UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Bedič

Emulacija študentske izkaznice NFC v pametnem telefonu

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Mira Trebar

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Razvoj tehnologij na področju komunikacije kratkega dosega (NFC-Near Field Communication) je omogočil široko uporabo kartic NFC za enostavno elektronsko identifikacijo. Razširjenost pametnih telefonov z vgrajenim modulom NFC ponuja nove rešitve z emulacijo oziroma direktno izvedbo enakih funkcionalnosti na sami napravi. Kandidat naj v diplomskem delu razišče možnosti za izvedbo kartice NFC na pametnem telefonu z operacijskim sistemom Android. Razvije in implementira naj mobilno aplikacijo tako, da prikaže vse podatke, ki so sedaj natisnjeni na študentski izkaznici NFC izdani na Univerzi v Ljubljani. Predlagana rešitev naj omogoča preverjanje istovetnosti v podatkovni bazi na strežniku. Za preverjanje in analizo delovanja naj realizira še mobilno aplikacijo za branje unikatne identifikacijske številke (UID) izkaznice in spletno aplikacijo s testno bazo študentov.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Matej Bedič, z vpisno številko **63070032**, sem avtor diplomskega dela z naslovom:

Emulacija študentske izkaznice NFC v pametnem telefonu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu prek univerzitetnega spletnega arhiva.

V Ljubljani, dne 6. marca 2015 Podpis avtorja:

Zahvaljujem se mentorici doc. dr. Miri Trebar, za mentorstvo in pomoč pri izdelavi diplomske naloge. Prav tako bi se zahvalil družini in prijateljem, ki so me podpirali v času izdelave diplomske naloge in celotnega študija.

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
AES	Advanced Encryption Standard	Napredni šifrirni standard
AID	Application Identification Number	Identifikacijska številka aplikacije
HCE	Host-based Card Emulation	Emulacija kartice
JSON	JavaScript Object Notation	Objektna notacija Javascript
MVC	Model View Controller	Model pogled krmilnik
NFC	Near Field Communication	Komunikacija kratkega dosega
P2P	Peer to Peer	Vsak z vsakim
REST	Representational state transfer	Predstavitveni prenos stanja
RFID	Radio Frequency Identification	Radiofrekvenčna identifikacija
SDK	Software Development Kit	Nabor razvojnih orodij
SQL	Structured Query Language	Strukturiran poizvedovalni jezik
UID	Unique Identification Number	Enolična identifikacijska številka
URI	Uniform Resource Indetifier	Enotni označevalnik vira

Povzetek

V diplomski nalogi smo predstavili izvedbo identifikacijske kartice v pametnem telefonu. Razvili smo mobilno aplikacijo, ki omogoča emulacijo ali posnemanje študentske izkaznice z vgrajenim NFC-čipom. Študent bi lahko namesto plastične kartice NFC, ki je trenutno v uporabi na Univerzi v Ljubljani, uporabljal svoj pametni telefon za preverjanje statusa in potrjevanje istovetnosti na predavanjih ali vajah, si izposodil knjigo v knjižnici ali imel dostop do določenih prostorov fakultete. Opisana je idejna zasnova ter tudi razvoj in uporaba aplikacije, ki deluje na napravah z operacijskim sistemom Android 4.4 KitKat ali novejšim. Za izvedbo zastavljene naloge smo uporabili vse bolj razširjeno tehnologijo NFC. Aplikacija je bila razvita s pomočjo programskega jezika Java in orodjem Android SDK v razvojnem okolju Android Studio. Za testiranje in prikaz delovanja sta bili razviti še dve dodatni aplikaciji: mobilna za branje kartic NFC in spletna s testno bazo podatkov študentov.

Ključne besede: Android, NFC, študentska izkaznica, emulacija, mobilna aplikacija

Abstract

In this thesis, we present the implementation of the identification card in a smartphone. We have developed a mobile application that allows emulation of a student card with NFC chip. Thus, instead of plastic NFC card, which is currently in use at the University of Ljubljana, student could use his smartphone for authentication on lectures or tutorials, borrow a book in the library or have access to certain premises of the Faculty. The thesis describes conceptual design and implementation of an application that operates on devices with Android 4.4 KitKat or later. To carry out the task ahead, we used increasingly widespread NFC technology. The application was developed using the programming language Java and the Android SDK tools in the development environment Android Studio. For testing and demonstration of the application two additional applications were developed: a mobile application that reads NFC cards and a web application with a test database of students.

Key words: Android, NFC, Student Id Card, emulation, mobile application

Poglavje 1 Uvod

Razvoj informacijsko komunikacijskih tehnologij in mobilnih naprav prinaša uporabnikom številne rešitve, ki mu poenostavijo različne aktivnosti. Posledično je čedalje bolj priljubljeno in celo zaželeno, da imamo čim večjo količino podatkov shranjenih na svojih pametnih mobilnih telefonih, med drugim tudi podatke o osebnih bančnih, plačilnih in kreditnih karticah. Naložene imamo aplikacije, s katerimi si lahko v sodobnem svetu tehnologij olajšamo vsakodnevno plačevanje računov, nakup vozovnic, nakupi v spletnih trgovinah in še veliko drugih aktivnosti. Porodila se nam je ideja, da bi lahko tudi študenti imeli na voljo aplikacijo s podatki, ki so sedaj zapisani na študentski izkaznici.

Študentska izkaznica je javna listina, s katero študentka ali študent Univerze v Ljubljani izkazuje status študenta. Uporablja se za preverjanje statusa pri opravljanju izpitov, pri uveljavljanju študentskih bonitet, kot so študentski boni in študentski popusti, in pri izposoji knjig na fakulteti. Pri identifikaciji je treba priložiti tudi osebni dokument.

Dandanes ima skoraj vsak študent pametni mobilni telefon z operacijskim sistemom Android, Windows ali iOS. Večina je že opremljena s tehnologijo NFC (Near Field Communication). Novejše študentske izkaznice so opremljene z čipom NFC, kar pomeni, da je komunikacija kratkega dosega že v splošni rabi. Identifikacija študentov je v tem primeru zelo preprosta.

V diplomskem delu smo predstavili emulacijo študentske izkaznice NFC v pametnem telefonu. Opisan je postopek prenosa podatkov o študentu iz 'testnega' okolja na strežniku na pametni telefon, kako so podatki shranjeni in prikazani na zaslonu ter kako se študentska izkaznica na telefonu uporablja. Razvili smo dve mobilni aplikaciji, kjer prva emulira študentsko izkaznico NFC, druga pa zagotavlja branje podatkov z emulirane kartice. Za izvedbo celovitega testiranja smo razvili še spletno aplikacijo in definirali bazo podatkov, v kateri so shranjeni testni podatki študentov.

Poglavje 2 Teoretične osnove

Za izvedbo in preverjanje delovanja študentske izkaznice NFC v pametnem telefonu so bile uporabljene različne tehnologije. Aplikacija za emulacijo ali posnemanje kartice in aplikacija za branje njenih podatkov sta razviti na platformi Android. Uporabljata tehnologijo NFC za prenašanje podatkov. Napisani sta v programskem jeziku Java, v razvojnem okolju operacijskega sistema Android.

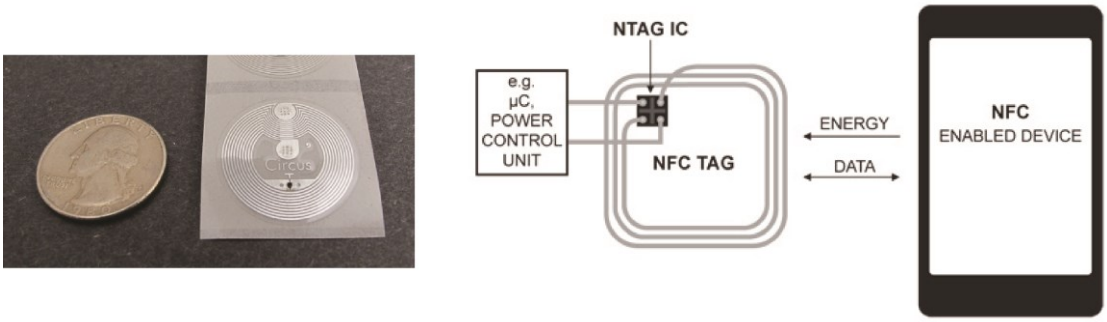
Spletna aplikacija s podatki študentov, ki so bili uporabljeni pri testiranju, je bila razvita na platformi ASP.NET, v okolju Visual Studio. Gostovanje spletne strani in baze podatkov je izvedeno na platformi za razvoj in gostovanje aplikacij v oblaku Microsoft Azure.

2.1 Near Field Communication

Near Field Communication (NFC) je tehnologija brezžičnega prenosa podatkov na razdalji do 10 cm ali krajše povedano komunikacija kratkega dosega. Omogoča enostavno in varno izmenjavo podatkov med dvema elektronskima napravama [5]. NFC sta leta 2002 predstavili podjetji Philips in Sony. Kmalu za tem ga je European Computer Manufacturers Association (ECMA) sprejela kot standard. Enako sta naslednje leto storili še International Organization for Standardization (ISO) in International Electrotechnical Commission (IEC). Nato pa je bila leta 2004 ustanovljena neprofitna družba NFC Forum, ki aktivno skrbi za to, da se tehnologija zelo hitro širi [1].

NFC uporabniku omogoča brezžične transakcije, dostop do digitalnih vsebin in povezavo elektronskih naprav z enim dotikom [11]. Lahko bi rekli, da je NFC eno od razvojnih področij delovanja tehnologije RFID (Radio Frequency Identification). RFID je tako kot NFC brezžična komunikacijska tehnologija, ki deluje prek radijskih valov [1]. Namenjena je za izmenjavo podatkov med RFID-bralnikom in značko. Značka je enostavna pasivna naprava brez lastnega napajanja, na katero lahko shranimo manjšo količino podatkov. Vsebuje silicijev čip, ki omogoča, da sprejema in odgovarja na signale naprave RFID (zahteve branja in pisanja). Tudi z napravo NFC lahko beremo značke (slika 1). Tako kot značka je tudi kartica NFC pasivna naprava in deluje po enakem principu.

RFID deluje na nizkih (Low Frequency: 125–134 kHz), visokih (High Frequency: 13.56 MHz) in ultra visokih (Ultra High Frequency: 400–930 MHz) frekvencah elektromagnetnega spektra, v nekaterih drugih primerih tudi na ravni mikrovalov. NFC uporablja glavne elemente standarda za tehnologijo brezkontaktnih kartic ISO/IEC 14443 A&B in deluje izključno na frekvenci 13,56 MHz. S tehnologijo NFC lahko prenesemo večjo količino podatkov kot z RFID, s hitrostjo do 424 kbit/s, medtem ko se pri RFID doseže največja hitrost do 200 kbit/s. Pri NFC poteka komunikacija na razdalji, ki meri največ 4 cm, medtem ko lahko promet RFID poteka na razdalji do 20 cm za pasivne naprave in do 400 cm za aktivne [1].



Slika 1: značka NFC [12] in potek komunikacije med njimi in napravo NFC [26].

NFC deluje na tri načine [1]:

- **Bralno/pisalni način** (ang. reader/writer mode) je namenjen komunikaciji naprave NFC z značko NFC (slika 1) za branje z nje ali pisanje nanjo. V obeh primerih vzpostavi komunikacijo naprava NFC, saj značke nimajo lastnega napajanja. Podatek, ki je zapisan na znački, je lahko karkoli. Najpogosteje je na njej poljubno besedilo ali spletni naslov. Ko naprava NFC prebere značko v bralnem načinu, se v brskalniku na tem naslovu odpre spletna stran.
- **Način P2P** (ang. peer-to-peer mode) omogoča prenašanje podatkov med dvema napravama NFC (tako deluje Android Beam). Tukaj gre za izmenjavo med aktivnimi napravami z lastnim napajanjem. Napravi si lahko izmenjata vrsto različnih podatkov, kontaktne številke iz imenika, tekstovne datoteke, slike in drugo.
- **Emulacija ali način posnemanja kartice** (ang. card emulation mode) omogoča, da se naprava obnaša kot brezkontaktna pametna kartica NFC. Največkrat se ta način uporablja za emuliranje kreditnih ali debetnih kartic in kartic zvestobe. Naprava, na kateri je emulirana kartica NFC, ne uporablja svojega napajanja za delovanje (se obnaša kot pasivna naprava). To pomeni, da naprava NFC, s katero beremo podatke na kartici, pošlje radiofrekvenčni signal, ta pa sproži delovanje aplikacije, ki emulira kartico NFC.

Varnost podatkov pri emulaciji kartice NFC

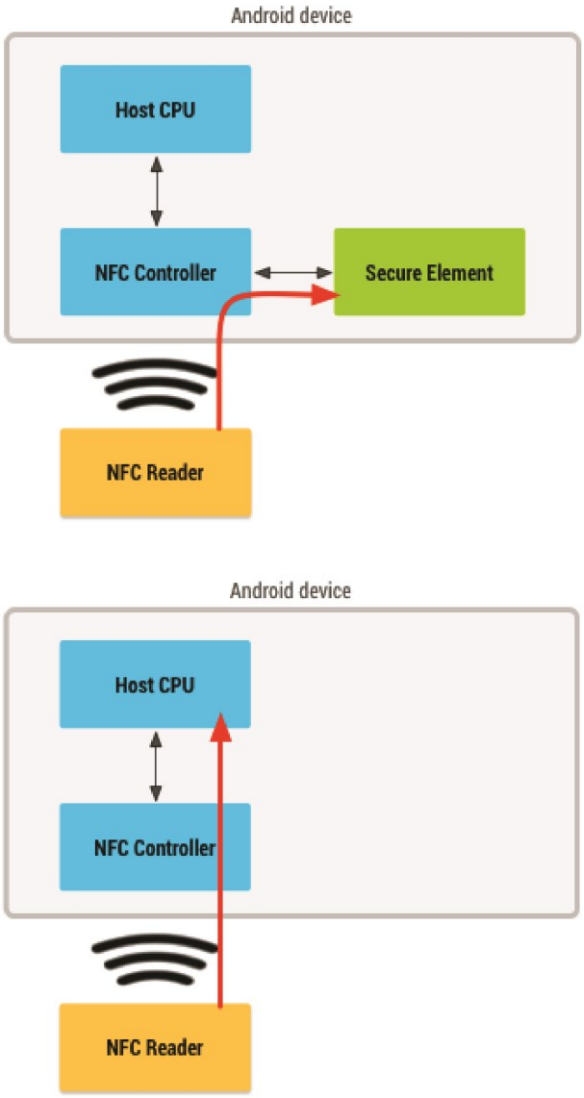
Za emulacijo kartice NFC na napravah Android potrebujemo varen prenos podatkov, saj nočemo, da kdorkoli kopira naše podatke in nam tako rekoč ukrade kartico. Emulirane kartice na pametnem telefonu, ki imajo občutljive, zasebne podatke (na primer bančne kartice), ne sme prebrati kar vsak bralnik kartic NFC. Prav tako se ne sme zgoditi, da bi kreditno kartico lahko prebral vsak pametni telefon z aplikacijo za branje kartic NFC ali značk in imel dostop do občutljivih podatkov.

Podatki, ki jih sprejme naprava, pridejo do krmilnika NFC (ang. *NFC Controller*) [8]. Od tam se prenesejo do procesorja v napravi, da lahko trenutno delujoča aplikacija izvede potrebne operacije.

Za varno kriptiranje podatkov, preden pridejo do procesorja v telefonu ali gredo iz njega in niso prosto dostopni bralnim napravam NFC, poskrbi varnostni element (ang. *Secure Element*). Verzije operacijskega sistema Android, ki so starejše od 4.4 KitKat, za varnost prenosa podatkov potrebujejo fizični element, kot so na primer kartica SIM, micro SD-kartica ali vgrajen varnostni element:

- Če uporabimo zunanji varnosti element (kartica SIM) moramo to narediti v sodelovanju s ponudnikom mobilnih storitev (Simobil, Telekom itd.), da dobimo dostop do varnostnega elementa. To nas omejuje, saj ni nujno, da se vsi ponudniki strinjajo s takšno uporabo. Tako aplikacije, ki bi uporabljala ta varnostni element, ne bi mogli uporabljati vsi, lahko bi jih samo tisti, katerih ponudnik je dovolil uporabo.
- Uporabimo lahko tudi kartico micro SD, vendar to pomeni, da potrebujemo posebno kartico micro SD samo za našo aplikacijo, kar spet ni najboljša rešitev, saj ima večina pametnih telefonov samo eno režo za kartico micro SD ali pa je sploh nima.
- Nekateri pametni telefoni imajo Secure Element fizično vgrajen v samo napravo, pri teh je za dostop do njega treba prositi izdelovalca. Vendar je takih pametnih telefonov premalo in za aplikacijo, ki bi jo uporabljalo veliko ljudi, ne pride v poštev.

Od verzije operacijskega sistema Android 4.4 KitKat naprej je ta težava rešena programsko s HCE (Host-based Card Emulation), kar pomeni, da za varen prenos podatkov do procesorja ne potrebujemo nobenega dodatnega elementa (slika 2).



Slika 2: Delovanje NFC z uporabo Secure Elementa zgoraj in HCE spodaj [8].

Delovanje HCE

Arhitektura HCE v operacijskem sistemu Android temelji na komponenti storitev (ang. HCE Service). Glavna prednost storitve je, da lahko teče v ozadju, kar pomeni, da uporabniku ni treba zagnati aplikacije, da bi lahko uporabil emulirano kartico na telefonu [8]. Ta deluje tako, da se ob kontaktu naprave in bralnika kartic NFC požene storitev HCE in opravi svojo nalogo. Ni potrebe po dodatnem grafičnem vmesniku ali zaganjanju aplikacije. Vse se opravi avtomatično. Lahko pa se obvesti uporabnika, da je bila storitev uspešno zaključena oziroma kartica prebrana.

Aplikacija za emulacijo kartice mora vedeti, ali lahko požene storitev HCE oziroma katero pognati, če jih imamo več. Uporablja se standard ISO/IEC 7816-4: ta opisuje način izbire

aplikacije na podlagi identifikacijske številke aplikacije (Application ID - AID) [9]. AID je šestnajstiška številka do velikosti 16 bytov. Pri emulaciji kartic za že obstoječe infrastrukture bralnikov kartic NFC so številke AID dobro znane in javno registrirane (na primer številke AID za plačilna omrežja, kot sta Visa in MasterCard).

Če želimo ustvariti novo infrastrukturo bralnika za aplikacijo, moramo registrirati svoj AID. Postopek registracije je opisan v standardu ISO/IEC 7816-5.

Mogoče je tudi, da je za zagon določene aplikacije treba določiti skupino AID. Poskrbeti je treba, da operacijski sistem ve, katere številke AID spadajo v skupino. Pomembno je tudi, da posamezna številka AID v skupini ne pripada kateri drugi aplikaciji.

Vsako skupino AID lahko vključimo v kategorijo. To Androidu omogoča, da združi storitve HCE v kategorije in tako dovoli uporabniku, da določi, katera kategorija je privzeta, saj navadnemu uporabniku AID-številke ne pomenijo nič. Android 4.4 podpira dve kategoriji CATEGORY_PAYMENT (ta vključuje standardizirane plačljive aplikacije) in CATEGORY_OTHER (vključuje vse druge aplikacije HCE).

Format sporočil NDEF

Podatki, ki se prenašajo med napravami NFC, so zapisani v formatu NDEF (ang. *NFC Data Exchange Format*). Vsako sporočilo NDEF (ang. *NDEF Message*) lahko vsebuje enega ali več zapisov (ang. *NDEF Records*). Sporočila se prenašajo v binarni obliki, kjer je vsak zlog sporočila točno določen. Vsak zapis vsebuje tip zapisa, unikaten identifikator, dolžino zapisa in same podatke [1]. Obstaja več dobro poznanih tipov zapisov NDEF, ki se uporabljajo v napravah NFC:

- **Zapis preprostega besedila** (ang. *Simple Text Records*) vsebuje poljubno vsebino za pošiljanje iz ene naprave v drugo. Po navadi ne vsebuje navodil za ciljno napravo. Vsebuje pa metapodatke z informacijami o jeziku napisanega besedila in kodiranju znakov.
- **Zapis URI** (ang. *URIs*) vsebuje spletni naslov. Po prejetju zapisa URI ciljna naprava odpre spletno stran z aplikacijo, ki jo lahko prikaže spletni brskalnik.
- **Pametni plakat** (ang. *Smart Posters*) lahko vsebuje dodatne informacije, ki so lahko v različnih oblikah, zapis URI ali besedilo, ki dodatno opisuje plakat. Ciljna naprava ob prejetju plakata lahko odpre različne aplikacije, to je odvisno od vsebine zapisa.
- **Podpis** (ang. Signature) je zagotovilo, da so podatki verodostojni.

2.2 Android Studio, Android SDK in SQLite

Android je ena najpopularnejših mobilnih platform na svetu, ki se uporablja tudi na mobilnih telefonih [1]. Je predelana verzija operacijskega sistema Linux z integriranim programskim vmesnikom, ki uporablja programski jezik Java. Android je v lasti Open Handset Alliance in je popolnoma odprtokodni sistem. Trenutno projekt vodi Google in večina kode je v obtoku pod licenco 'Apache License'.

Android Software Development Kit (Android SDK) je bogat nabor orodij, ki vključuje razhroščevalnik (ang. debugger), knjižnice, posnemovalnik (ang. emulator), dokumentacijo, primere kode in vodene vaje (ang. tutorials) [1]. Omogoča preprost razvoj aplikacij z uporabo Android Studia (uradna platforma za Android).

Android Studio je uradno integrirano razvojno okolje (IDE - Integrated development environment) za razvoj aplikacij Android, ki temelji na okolju Java IntelliJ IDEA [13]. Android Studio ponuja:

- prilagodljiv sistem, grajen na osnovi Gradle,
- različne možnosti razvoja projektov in več datotek .apk,
- predloge kode, ki nam pomagajo graditi skupne značilnosti aplikacij,
- bogat urejevalnik postavitve grafičnih elementov (način povleci–spusti),
- orodje Lint za lovljenje različnih napak (napačna verzija, napačna uporaba metod itd.),
- ProGuard in podpisovanje (kriptiranje) aplikacij,
- vgrajeno podporo za Google Cloud Platform, s katerim je zelo lahko vgraditi podporo za pošiljanje sporočil z Google Cloud Messaging in druge Googlove storitve,
- in še veliko več.

SQLite je vgrajena podatkovna baza SQL [14]. V nasprotju z večino podatkovnih baz SQLite nima ločenega strežniškega procesa. To pomeni, da za svoje delovanje ne potrebuje strežnika, saj se nahaja na napravi. Podatkovna baza bere in piše neposredno na trdi disk. Cela baza z vsemi tabelami, pogledi in sprožilniki je vključena v eno datoteko na disku. Ta datoteka je prenosljiva tako med različnimi operacijskimi sistemi, kot tudi med 32- in 64-bitnimi sistemi ali arhitekturami tankega in debelega konca. Zato je SQLite zelo priljubljena in uporabljena podatkovna baza.

2.3 Java

Aplikacije za operacijski sistem Android so pisane v programskem jeziku Java [1]. Java je objektno usmerjen programski jezik, ki je oblikovan tako, da je neodvisen od platforme. Leta 1995 ga je predstavilo podjetje Sun Microsystems.

Javanski programi so napisani v datoteki s končnico .java. Ti niso prevedeni v strojni jezik naprave (na kateri se izvajajo), ampak se prevedejo in izvajajo na prevajalniku. Odvisni so od sistema, na katerem se izvajajo. Pretvorniki na osebnih računalnikih se imenujejo Java Virtual Machines (JVMs). Preden se požene program v datoteki s končnico .java, jo je treba s prevajalnikom prevesti in pretvoriti v datoteko s končnico .class, ki vsebuje bitno kodo. Program v bitni kodi je nato izveden na virtualni napravi.

2.4 Microsoft ASP.NET, Visual Studio in Microsoft Azure

.NET je razvojno orodje za razvijanje aplikacij za operacijski sistem Windows, telefone Windows, strežnike Windows in Microsoft Azure. Podpira razvoj aplikacij naslednje generacije in spletnih storitev XML. Vključuje knjižnice razredov, vmesnikov, tipe vrednosti. Orodje .NET ponuja vodljivo izvajalno okolje, preprost razvoj in namestitev ter integracijo z različnimi programskimi jeziki, na primer Visual Basic in Visual C# [24].

Visual Studio 2012 je integrirano razvojno orodje (IDE), ki se uporablja za razvoj spletnih aplikacij in spletnih storitev. Razvijajo se lahko aplikacije z grafičnim vmesnikom (GUI) ali brez njega. Podpira vrsto programskih jezikov C#, Visual Basic, JavaScript, C++, F#, XML/XSLT, HTML/XHTML in CSS.

ASP.NET MVC je orodje za razvoj spletnih aplikacij. Zgrajen je na osnovi orodja .NET. MVC (Model-View-Controller) je v nasprotju s spletnimi obrazci (ang. Web Forms) veliko bolj vodljiv, saj imamo popolno kontrolo nad vsakim delom aplikacije (prikaz podatkov, obdelava podatkov, pretok podatkov) [2]. MVC aplikacijo razdeli na tri dele:

- Model – pomeni jedro poslovne logike in podatke. Je objekt, ki predstavlja podatke.
- Pogled – njegova naloga je, da model predstavi v vizualni obliki. To največkrat pomeni generiranje kode HTML, ki se prikaže v spletnem brskalniku.
- Krmilnik – kontrolira logiko aplikacije in deluje kot koordinator med modelom in pogledom.

MICROSOFT AZURE

Microsoft Azure je odprta in fleksibilna platforma, ki razvijalcem in IT profesionalcem omogoča, da razvijajo in upravljajo aplikacije, ki se izvajajo v Microsoftovih podatkovnih centrih na različnih lokacijah po svetu [23].

Omogoča vrsto storitev, ki so povezane z računalništvom v oblakih. Omogoča gostovanje spletne strani, podatkovne baze, mobilne spletne aplikacije in še veliko več. Podpira tako odprtokodne tehnologije kot celotno ogrodje .NET, Windows in strežnik SQL.

Aplikacije s spletnim vmesnikom, ki so dostopne prek brskalnika, delujejo na principu odjemalec–strežnik in so razvite na tehnologiji ASP.NET, se lahko zelo hitro prenesejo na platformo Windows Azure. Pri tem se nabor osnovnih tehnologij ne spremeni, saj se taka aplikacija še vedno izvaja v okolju Windows Server na tehnologiji .NET.

REST in RESTful

REST (Representational State Transfer) je programska arhitektura aplikacije, zgrajena glede na način, kako so podatki predstavljeni, dostopni in spremenjeni na spletu [7]. V arhitekturi REST so podatki in funkcionalnosti predstavljeni kot viri, do katerih dostopamo z naslovi URI (Uniform Resource Identifiers). Dostop je izveden z uporabo preprostih, dobro definiranih operacij. Arhitektura REST je v osnovi odjemalec–strežnik arhitektura in je oblikovana tako, da uporablja komunikacijski protokol brez stanj, največkrat http (Hypertext Transfer Protocol).

Spletna storitev RESTful je spletna aplikacija, zgrajena na arhitekturi REST. Do virov dostopa prek naslovov URI in uporablja štiri glavne metode http za upravljanje virov:

- GET (se uporablja za pridobivanje vira),
- POST (se uporablja za ustvarjanje vira),
- PUT (namenjena urejanju vira),
- DELETE (se uporabi za brisanje vira).

Viri se prenašajo oziroma pošiljajo v formatu JSON ali XML.

Format JSON

JSON (JavaScript Object Notation) je format zapisa za izmenjavo podatkov, na primer med spletno in mobilno aplikacijo. Izhaja iz programskega jezika JavaScript [25]. Preprost je za branje in pisanje ter za razčlenjevanje in generiranje. Temelji na dveh strukturah:

- Zbirka ime/vrednost. V različnih jezikih je to realizirano kot objekt, zapis, slovar.

- Urejen seznam vrednosti. V večini jezikov je realiziran kot tabela, seznam, sekvenca.

Poglavje 3 Identifikacijska kartica NFC

Kartice, ki vsebujejo čip NFC, so dandanes čedalje bolj razširjene. Uporabljajo jih lahko uslužbenci za vstop v poslopje podjetja, kjer delajo, ali na primer za plačevanje v mestnem potniškem prometu. Vedno več pametnih telefonov omogoča uporabo tehnologije NFC. Zakaj ne bi za zgoraj omenjena primera in še več drugih uporabljali pametnih telefonov namesto kartic?

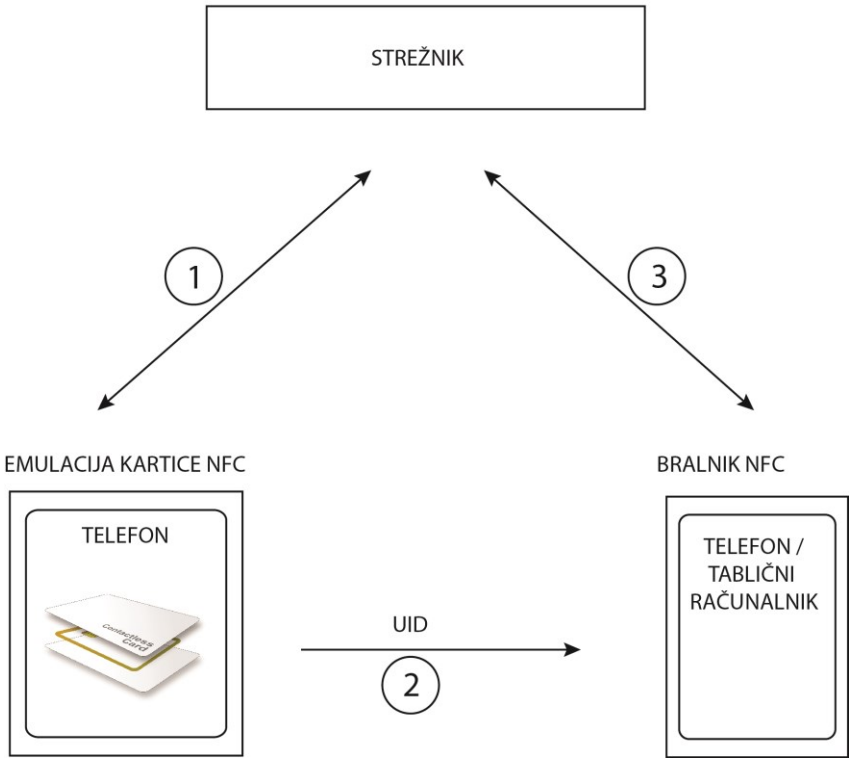
Problem, ki ga želimo rešiti, je namestitev študentske izkaznice v pametni telefon. Treba je določiti, kateri podatki bodo prikazani na zaslonu, kateri bodo dostopni s tehnologijo NFC, kako se podatki prenesejo na pametni telefon in kako bodo shranjeni na pametnem telefonu.

Za prikaz predlagane rešitve bomo razvili tri aplikacije:

- mobilno aplikacijo, ki emulira študentsko izkaznico NFC (»Emulacija kartice NFC«),
- spletno aplikacijo z bazo podatkov in
- mobilno aplikacijo, ki deluje kot bralnik kartic NFC (»Bralnik NFC«).

Slika 3 prikazuje povezanost aplikacij in prenos podatkov med njimi:

1. Prvi korak vključuje pridobivanje podatkov študentske izkaznice na pametni telefon. Podatki bodo potovali v obe smeri. Aplikacija Emulacija kartice NFC bo poslala zahtevo po podatkih. Če bo imela zahteva pravilne podatke oziroma parametre, bo spletna aplikacija poslala podatke zahtevane študentske izkaznice aplikaciji.
2. Pri drugem koraku bo potekalo preverjanje identitete študentov. Povezava bo enosmerna in bo usmerjena proti aplikaciji Bralnik NFC. Ta bo preverjal identiteto s pomočjo dobljenega UID, ki ga bo poslala Emulacija kartice NFC.
3. Pri tretjem koraku bo Bralnik NFC želel preveriti, ali je študent s pripadajočo izkaznico zapisan v bazi podatkov na strežniku. Če je vpisan, se bodo podatki prenesli na napravo in prikazali na zaslonu. Podatki se bodo prenašali v obe smeri.



Slika 3: Povezanost aplikacij Emulacija kartice NFC, Bralnik NFC in spletne aplikacije z bazo podatkov na strežniku.

3.1 Emulacija kartice NFC

Študentske izkaznice Univerze v Ljubljani, ki imajo vgrajen čip NFC, se uporabljajo kot identifikacijske kartice, s katerih se podatki berejo s pomočjo tehnologije NFC. V spominu so shranjeni kriptirani podatki o ustanovi in študentu ter enolični identifikator (UID). Za običajno preverjanje statusa študenta, pa ima izkaznica vse podatke tudi natisnjene na obeh straneh.

3.1.1 Študentska izkaznica NFC

Študentska izkaznica ima vgrajeno pasivno napravo NFC MIFARE DESFire EV1 MF3ICD81 z integriranim vezjem in jo izdeluje NXP [6]. Čip integriranega vezja temelji na standardu ISO/IEC 14443 tipa A, ki opisuje delovanje pametnih kartic. Zmogljivost kartice za hranjenje podatkov je 8KB. Za zaščito podatkov uporablja kriptirni algoritem DES/3DES ali AES [4]. Hitrost prenosa lahko sega do 848 KBit/s. Unikatni identifikator UID (Unique Identification Number) uporablja 14-mestno šestnajstiško število. V kartici je UID zapisan v del spomina, iz katerega je mogoče le brati, ne pa tudi pisati.

Izkaznica ustreza standardu za identifikacijske izkaznice ISO/IEC 7810 format ID-1, to je 85,60mm x 53,98mm x 0,76mm [3]. Izkaznica ima unikatno oznako za slepe, da lahko uporabnik po znaku prepozna posamezno izkaznico. Oblikovana je v skladu s celotno grafično podobo UL. Vsi opisi polj na izkaznici so zapisani v slovenskem in angleškem jeziku (slika 4).

Podatki, ki so vidni na prvi strani študentske izkaznice:

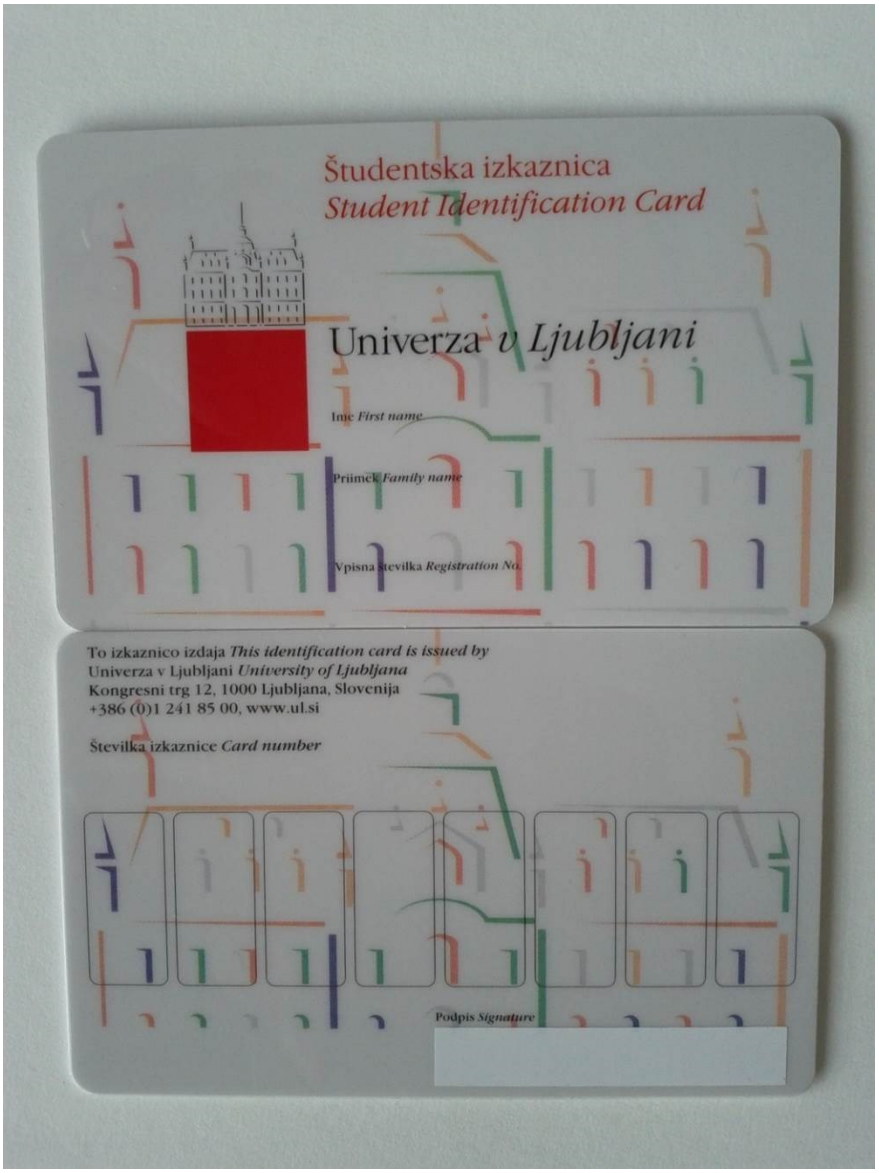
- Naziv izkaznice: Študentska izkaznica
- Naziv univerze: Univerza v Ljubljani
- Logotip univerze
- Ime
- Priimek
- Vpisna številka

Na hrbtni strani izkaznice so podatki:

- o izdajatelju,
- prostor za podpis imetnika,
- prostor za nalepke za izkaz veljavnosti,
- številka izkaznice.

V čipu na izkaznici se nahajajo podatki:

- o imetniku izkaznice (ime, priimek, vpisna številka),
- o izkaznici (UID izkaznice in naziv izkaznice),
- prostor za dodatno storitev UL,
- prostor za dodatne zunanje storitve.



Slika 4: Študentska izkaznica s čipom NFC.

3.1.2 Emulirana študentska izkaznica NFC

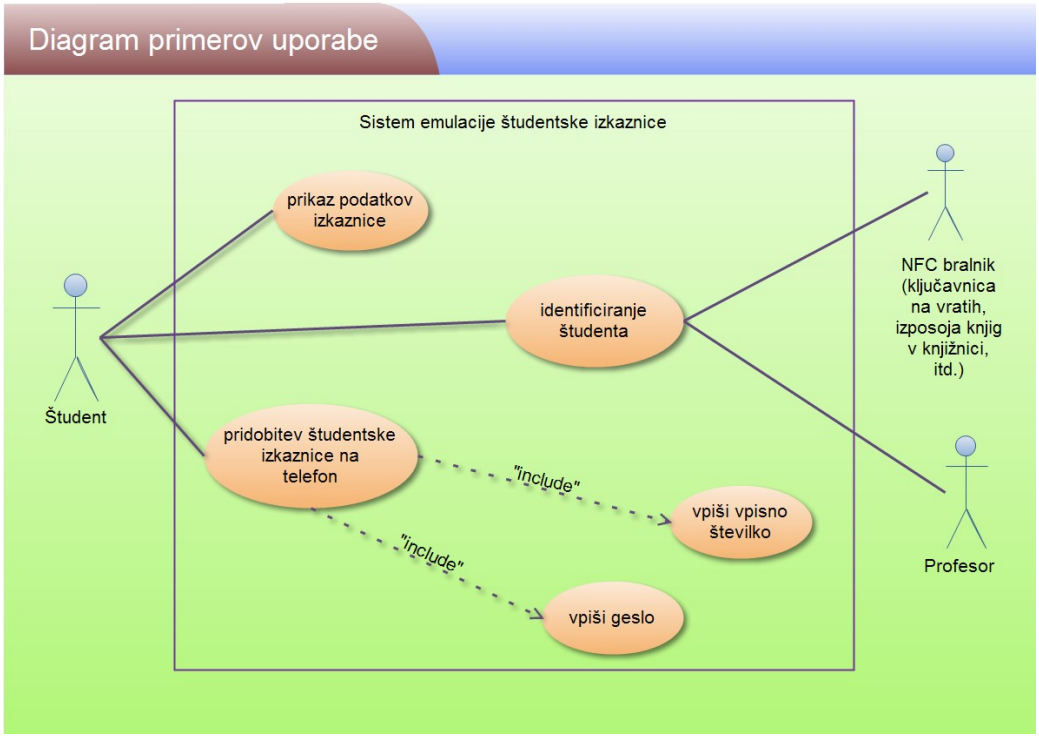
Naša rešitev bo delovala v enem od treh opisanih načinov komuniciranja prek NFC, emulacija kartice NFC. Ta omogoča pametnemu telefonu, da se obnaša kot brezkontaktna kartica NFC za identificiranje ali opravljanje različnih storitev (na primer plačevanje). Aplikacija bo razvita za operacijski sistem Android. Sicer je tehnologija NFC prisotna tudi v mobilnih napravah Apple in Windows, vendar telefoni Windows niso tako široko uporabljeni, Apple pa je uporabo NFC omejil samo na plačljivo platformo Apple Pay.

Vsi podatki, ki so vidni na študentski izkaznici, bodo vidni tudi v naši aplikaciji. Med napravama NFC se bo prenašal UID. V aplikaciji je ta podatek zaščiten in ga bo mogoče, ko se bo enkrat kartica naložila na mobilno napravo, samo brati.

Na sliki 5 je prikazan diagram primerov uporabe emulirane študentske izkaznice. Ta prikazuje glavne funkcionalnosti aplikacije, ki emulira študentsko izkaznico:

- prikaz podatkov izkaznice,
- identifikacija študenta,
- pridobitev študentske izkaznice na telefon.

Funkcionalnosti »vpiši vpisno številko« in »vpiši geslo« sta vsebovani v glavni funkcionalnosti »pridobitev študentske izkaznice na telefon«.



Slika 5: Diagram primerov uporabe.

3.1.3 Pridobitev študentske izkaznice na telefon

Pristojni organ za izdajo študentske izkaznice je Univerza v Ljubljani. Referati za študentske zadeve članic (akademije in fakultete) posredujejo rektoratu UL podatke o prvič vpisanih študentih. Rektorat jih posreduje izdelovalcu izkaznic. Le-ta izdelane izkaznice pošlje nazaj rektoratu, ta pa naprej članicam [3].

Postopek zapisa študentke izkaznice na pametni telefon, ki je implementiran v diplomski nalogi, bi bil drugačen:

- Po vpisu študenta v bazo podatkov bi si študent naložil s spleta aplikacijo na svoj pametni telefon.
- Po namestitvi aplikacije bi moral študent za pridobitev izkaznice vpisati na telefon svojo vpisno številko in geslo, ki bi ju dobil v referatu fakultete. Za to bi moral imeti vzpostavljeno spletno povezavo.
- Ob kliku na potrditveni gumb bi se vpisana podatka poslala na strežnik s spletno aplikacijo, ki upravlja podatkovno bazo študentov (korak 1 na sliki 3).
- Prejeta podatka se preverita v bazi podatkov. Če obstajata in sta pravilna, strežnik iz baze podatkov pošlje podatke o študentu nazaj na napravo.
- Ob prevzemu podatkov bi aplikacija avtomatično sporočila, ali so bili vsi podatki pravilno prejeti in shranjeni v telefon. Ob pravilnem prevzemu bi se podatki shranili v telefon in postopek prevzema bi bil končan. Če bi prišlo do napake, bi aplikacija ponovno zahtevala vnos vpisne številke in gesla študenta.

3.1.4 Prikaz podatkov

Aplikacija bo omogočala prikaz podatkov študenta na zaslonu. Podatki, ki bodo v aplikaciji vidni, so:

- ime,
- priimek,
- vpisna številka,
- naziv univerze,
- logotip univerze,
- veljavnost izkaznice (ali študent ima status študenta).

V aplikaciji bo dodana tudi možnost shranjevanja in prikazovanja slike študenta, ki si jo bo lahko izbral sam. Seveda bo morala slika ustrezati vnaprej določenim zahtevam.

3.1.5 Shranjevanje podatkov

Za varnost podatkov na poti od krmilnika NFC do procesorja poskrbi HCE. Ko pa so enkrat podatki v procesorju, mora zanje poskrbeti operacijski sistem. Privzeti način shranjevanja podatkov v Androidu pomeni, da shranjenih podatkov določene aplikacije druge aplikacije ne vidijo. To seveda lahko spremenimo z uporabo razreda ContentProvider.

Čeprav aplikacije ne vidijo podatkov druga druge, se lahko do teh podatkov dokopljemo, če imamo na telefonu dostop do korenskega imenika (ang. root access). Tak dostop pomeni, da imamo vse upravljalske pravice telefona.

Pri shranjevanju imamo dve možnosti. Prva je, da so podatki shranjeni na telefonu, kjer so kriptirani, in za dostop do njih potrebujemo ustrezen ključ. Pri drugi možnosti pa imamo podatke shranjene zunaj naprave, v bazi podatkov ali oblaku, do katere dostopamo prek spleta. Tu moramo poskrbeti, da je povezava do baze varna. Zunaj telefona lahko shranimo podatke tudi na kartico SD.

Navadno so podatki, ki so shranjeni v datoteki, nevidni drugim aplikacijam v telefonu. Če imamo dostop do korenskega imenika telefona, morajo biti podatki kriptirani. To lahko storimo na primer s pomočjo razreda Cipher. Le-ta omogoča uporabo kriptografskih sistemov AES in RSA. Vprašanje je, kdo ima oziroma kje je shranjen ključ, ki je uporabljen za kriptiranje in dekriptiranje podatkov. Lahko ga shranimo v pomnilnik telefona, vendar to ni najbolj pametno, saj z dostopom do korenskega imenika lahko pridemo do ključa. Lahko ga shranimo na zunanjo kartico SD. To ni najbolj splošna rešitev, saj nimajo vsi mobilni telefoni te možnosti.

Ena od možnosti za kriptiranje bi bila zahtevati vpis gesla ob uporabi kartice od uporabnika in to geslo uporabiti za kriptiranje podatkov. Vendar bi to izničilo glavni namen storitve HCE, da se lahko promet NFC odvija v ozadju brez vpliva uporabnika.

Na telefon lahko shranimo podatke tako, da jih zapišemo v datoteko, ali pa ustvarimo svojo zasebno podatkovno bazo, kamor lahko shranimo strukturirane podatke v tabele. V naši aplikaciji je podatkov malo, saj bomo imeli na telefonu samo eno kartico NFC, kar pomeni, da zasebna podatkovna baza ni potrebna. Za prikaz delovanja smo ključ, uporabljen za kriptiranje in dekriptiranje, shranili v pomnilnik v telefonu.

3.2 Bralnik NFC

Ta aplikacija bo imela vlogo bralnika kartic NFC, katerega osnovna naloga bo preverjanje identitete študenta. Aplikacijo lahko uporablja profesor na telefonu ali tabličnem računalniku za preverjanje prisotnosti na izpitih, vajah in predavanjih.

Bralnik NFC mora sprožiti aplikacijo Emulacija kartice NFC. To pomeni, da podatkov kartice NFC ni mogoče prebrati z vsakim bralnikom oziroma aplikacijo za branje značk NFC na drugi mobilni napravi. Če bralnik ne pošlje pravih podatkov, potem tudi ne bo dobil odgovora.

Aplikacija se bo uporabila za prikaz, kako bi lahko na primer profesorji preverili verodostojnost študenta. Postopek preverjanja identitete bi potekal takole:

- Aplikacija za preverjanje študentov, Bralnik NFC, mora biti aktivirana.
- Napravo z emulirano kartico NFC bi približali napravi, na kateri se nahaja bralnik. Ta pošlje signal, v katerem je AID, s katerim povemo, katera aplikacija naj se zažene na napravi s kartico. Če je AID pravi, naprava s kartico pošlje signal, ki potrdi pravi AID.
- Zatem naprava s kartico pošlje UID študentske izkaznice.
- Če ima bralnik seznam študentov shranjen na napravi, potem dobljeni UID poišče v tem seznamu. Če seznama nima, se poveže z bazo podatkov na strežniku ter tam poišče študenta s poslano številko UID. Podatke tega študenta prenese iz baze podatkov in ga doda v svoj seznam študentov.
- Na bralniku se prikažejo podatki študenta.

Branje številke UID bo zasnovano enako kot pri že obstoječi aplikaciji. Ta aplikacija je bila razvita v diplomskem delu z naslovom Izkaznica NFC v postopku preverjanja študijskih obveznosti [10].

3.3 Spletna aplikacija in baza podatkov

Spletna aplikacija bo delovala kot vmesnik za dostop do testne baze podatkov študentov. V bazi bodo shranjeni osnovni podatki študentov:

- ime,
- priimek,
- vpisna številka,
- ali ima študent status ali ne in
- številka izkaznice (UID).

Namenjena bo pridobivanju podatkov študentske izkaznice in obema mobilnima aplikacijama Emulacija kartice NFC in Bralnik NFC.

V aplikaciji bo prikazan seznam vseh študentov. Omogočeno bo dodajanje novih študentov v bazo podatkov. Že vpisane študente v seznamu bo mogoče odstraniti ali jim prirediti podatke. Podatke posameznega študenta ali celoten seznam bomo lahko tudi izvozili v datoteko .csv. Podatkovna struktura baze podatkov je prikazana v tabeli 1.

Tabela 1: Podatki študentov v testni bazi podatkov.

ID študenta	**Ime**	**Priimek**	**Vpisna številka**	**Status študenta**	**Številka izkaznice (UID)**
0	Matej	Bedič	63070032	TRUE	28 8B 70 C9 05 40 DC
1	Jan	Pelko	63070123	TRUE	A0 98 45 48 AB BF 64
2	Špela	Krap	61080212	FALSE	5F 8E F6 A8 7F C4 4E
3	Janez	Novak	61081223	FALSE	F8 AA 41 8C BD BA 2F
4	Marija	Abunar	63071123	TRUE	13 1B AC BC DF 01 2C

Spletna aplikacija bo delovala kot spletna storitev RESTful [7]. S pomočjo te bo spletna aplikacija komunicirala z mobilnima aplikacijama Emulacija kartice NFC in Bralnik NFC. Mobilni aplikaciji bosta za pridobitev želenih podatkov iz baze pošiljali preprosto zahtevo, metodo GET. Opisana komunikacija je prikazana na sliki 3, koraka 1 in 3.

Po obeh povezavah se bodo s spletne aplikacije na mobilne aplikacije prenašali podatki v obliki formata JSON. Primer podatka, ki bi ga spletna aplikacija poslala pametnemu telefonu, na katerem je naložena aplikacija Emulacija kartice NFC:

{"studentID":1,"Name":"Jan","Surname":"Pelko","EnrolNumber":"63070123","StudentStatus":true,"UID":"a0984548abbf64" }

Poglavje 4 Razvoj aplikacij in testiranje

Razvoj smo začeli z miselnim vzorcem, kjer je prikazano, kako so vse tri aplikacije povezane (shema prikazana na sliki 3). Tukaj se je bilo treba odločiti, kateri podatki in v kakšni obliki bodo potovali od ene do druge aplikacije. Nato pa sta sledila načrtovanje aplikacij in programiranje. Pri vsaki aplikaciji smo začeli z razvojem uporabniškega vmesnika, nato smo definirali vse potrebne funkcionalnosti.

Začeli smo z razvojem aplikacije Emulacija kartice NFC. Sledili so razvoj spletne aplikacije ter postavitev strežnika in baze podatkov. Nazadnje smo se lotili še aplikacije Bralnik NFC.

4.1 Emulirana študentska izkaznica

V diplomski nalogi je bila razvita študentska izkaznica NFC na pametnem telefonu. Namesto da bi študenti uporabljali plastično kartico, bi lahko za preverjanje identitete uporabili kar svoj pametni telefon. Prav tako bi lahko svoje podatke prikazali na zaslonu svojega telefona.

Glavna funkcionalnost aplikacije Emulacija kartice NFC je pošiljanje UID študentske izkaznice napravi, ki želi izkaznico prebrati. To se mora zgoditi brez pomoči uporabnika, kar pomeni, da uporabniku ni treba zagnati aplikacije. Najprej pa moramo podatke izkaznice pridobiti na mobilno napravo. Aplikacija ima tudi možnost prikaza podatkov študentske izkaznice na zaslonu, zato smo razvoj začeli z uporabniškim vmesnikom. Ko so bili vsi elementi na zaslonu postavljeni, smo jim začeli dodajati funkcionalnosti. Nazadnje smo dodali storitev za komunikacijo med dvema napravama NFC.

4.1.1 Arhitektura aplikacije

Aplikacijo v Android Studiu sestavljajo trije deli:

- mapa *java*, ki vsebuje javanske razrede, v katerih je zapisana programska koda funkcionalnosti aplikacije,
- mapa *res*, v kateri so datoteke s končnico .xml, ki opisujejo in sestavljajo uporabniški vmesnik, in
- mapa *manifests*, ki vsebuje datoteko AndroidManifest.xml.

Mape so razdeljene v podmape, da je tako na pogled kot v sami kodi laže najti določeno datoteko.

V mapi *java* se nahajajo:

ena aktivnost (ang. Activity):

- MainActivity,

mapa *mainfragments* s fragmenti (ang.fragments):

- FragmentGetData,
- FragmentHome,
- FragmentStudentId,

mapa *fragmentaccesories* s fragmenti:

- FragmentChangeListener,
- FragmentDialogChangePofilePic,
- FragmentDialogExitApplication,

mapa *util*, ki vsebuje pomožne razrede:

- Crypt (vsebuje kodo za šifriranje podatkov),
- JSONParser (ta vsebuje metode za pridobitev podatkov, ki so zapisani v formatu JSON),
- Student (se uporablja za ustvarjanje objekta tipa Student, ki vsebuje podatke študentske izkaznice),
- Utilities (tu so zapisane pomožne metode za delo z nizi),

ter ena storitev (ang. service):

- StudentCardService.

Glavne funkcionalnosti aplikacije Android so aktivnosti. Te kažejo neko trenutno stanje aplikacije, kaj je prikazano in kaj se dogaja v ozadju. Vendar preveč aktivnosti tudi ni dobro, ker je potem zelo veliko komuniciranja med preveč komponentami, kar naredi aplikacijo slabo pregledno in težko za vzdrževanje. To lahko elegantno rešimo z uporabo fragmentov, ki so videti kot aktivnosti, a se obnašajo kot navadni prikazovalni elementi. Zato je z njimi tudi laže operirati glede prikazovanja vsebine na zaslonu in komunikacije med njimi.

Mapa *res* vsebuje:

- mapo *drawable*, v kateri so shranjene slike (na primer zagonska ikona), ter datoteke, s katerimi so določene lastnosti in obnašanje posameznih elementov uporabniškega vmesnika,
- mapo *layout*, ta vsebuje datoteke s končnico .xml, ki določajo razporeditev elementov uporabniškega vmesnika po zaslonu,
- mapo *values*, ki vsebuje datoteke, kjer so zapisane uporabljene barve (*colors.xml*), uporabljena besedila, na primer besedila gumbov (*string.xml*), in
- mapo xml, kjer se nahaja datoteka *aid_list.xml*, ki vsebuje podatke, na podlagi katerih se bo aplikacija odzvala na določen bralnik NFC ali ne.

V datoteki AndroidManifest.xml so opisani:

- pravila (kakšne tehnologije bo aplikacija uporabljala, na primer tehnologijo NFC, ali bo dostopala do spleta …),
- aktivnosti (za vsako aktivnost moramo določiti namen, na primer katera se bo prva prikazala ob zagonu aplikacije) in
- storitve (v našem primeru storitev emulacije kartice).

4.1.2 Uporabniški vmesnik

Začeli smo s podobo študentske izkaznice. Najpomembnejše je bilo, da so pregledno prikazani vsi podatki, ki so na trenutni kartici NFC. Odločili smo se, da bomo podatke prikazovali v portretnem načinu, saj mobilne naprave večino časa držimo v tem položaju. Določili smo, kako bi bili posamezni elementi (tekstovna polja, slike, ikone …) razporejeni po zaslonu. Vse informacije o tem so zapisane v datotekah, ki se nahajajo v mapi *res*.

Aplikacija je bila razvita za uporabo na mobilnih napravah, predvsem telefonih, lahko pa tudi na tabličnih računalnikih, ki imajo različno velike zaslone. Zato je razporeditev elementov na zaslonu v datotekah .xml določena z atributom utež (ang. weight). Ta nam pove, koliko prostora v omejenem območju neki element zasede. Recimo, da imamo v eni vrsti tri gumbe. Vsak gumb ima atribut uteži nastavljen na 1. To pomeni, da bo velikost vseh gumbov enaka oziroma bodo vsi gumbi zapolnili enako velik prostor v vrstici, saj so v razmerju 1:1:1. Če bi enemu gumbu spremenili vrednost atributa uteži na 2, bi bilo razmerje med gumbi 2:1:1, kar pomeni, da bi en gumb zavzel enkrat več prostora kot druga dva. Tako je postavitev elementov uporabniškega vmesnika podobna na vseh velikostih zaslonov mobilnih naprav.

Uporabili smo barve, ki so prisotne na sedanji študentski izkaznici. Ozadje je svetlo bež barve, na kateri se črno besedilo lepo vidi. Okoli elementov na zaslonu je obroba rdeče barve. Za gumbe in tekstovna polja smo uporabili rumeno-oranžno barvo z rdečo in zeleno obrobo. Priredili smo odtenek barv, da ne bi večji elementi preveč izstopali.

Gumbi so interaktivni. Ko je gumb neaktiven, je obrobljen z rdečo barvo, ko pritisnemo nanj, se obroba obarva zeleno. Niso vsi gumbi vedno uporabni oziroma njihova funkcionalnost ni vedno potrebna. Na primer, preden pridobimo podatke študenta na mobilno napravo in ustvarimo izkaznico ter shranimo podatke v pomnilnik, gumba za prikaz teh podatkov še ne moremo uporabljati. Zato smo v tem primeru gumb zasenčili in ga naredili neaktivnega, kar pomeni, da se ob pritisku nanj ne zgodi nič.

Tudi polja besedila so interaktivna, saj so, ko so prazna, podčrtana rdeče, ko pa je vstavljeno besedilo in ko izgubijo fokus, so podčrtana zeleno. Polje za vpis gesla je zaščiteno tako, da ni vidno vpisano besedilo. Vsak znak je zamenjan s piko. Če trikrat vtipkamo napačno geslo, se aplikacija zaklene. To pomeni, da so vnosna polja in gumb za potrditev neaktivni. Sočasno se prikaže obvestilo, naj se obrnemo na študentski referat ter zaprosimo za novo geslo.

4.1.3 Podatki

Podatke, ki se nahajajo na dejanski študentski izkaznici NFC, smo shranili v navadno datoteko. Potrebe po podatkovni bazi ni bilo, ker podatkov ni veliko in jih ni treba shranjevati v tabele. Podatki so shranjeni kot znakovni niz. Ta datoteka je nedostopna drugim aplikacijam ali samemu uporabniku zunaj aplikacije. Do nje lahko dostopamo samo, če imamo dostop root do datotečnega sistema. Čeprav so podatki nedostopni, jih aplikacija šifrira, preden jih shrani v datoteko. Tako so varni tudi, če bi dobili dostop do te datoteke. Vsakič, ko želimo podatke prikazati na zaslonu, se datoteka prebere in dešifrira.

Za šifriranje podatkov uporabljamo pomožni razred *Crypt*. V njem uporabljamo razred *Cipher* iz nabora vmesnikov Android API (Application programming interface), ki omogoča implementacijo kriptografskih šifer za šifriranje in dešifriranje besedila [15].

Razreda *Cipher* ne moremo pognati direktno, poklicati moramo konkretno izvedbo objekta razreda z metodo *getInstance*. Ob klicu te metode kot parameter podamo znakovni niz, ki določa, kateri algoritem se bo uporabil za šifriranje. V naši aplikaciji smo uporabili algoritem AES (Advanced Encryption Standard) algoritem [16]. AES je simetrični šifrirni algoritem, ki uporablja bloke velikosti 128 bitov, ter šifrirne ključe, katerih dolžina je 128, 192 ali 256 bitov. Ključ je shranjen v kodi aplikacije.

Ob zapisu podatkov izkaznice v datoteko se uporabi metoda *encrypt*, ob klicu katere podamo podatke v obliki znakovnega niza. Ta znakovni niz šifrira v nov znakovni niz, nato pa se le-ta shrani v datoteko.

Ob vsakem branju se uporabi metoda *decrypt*, kjer je parameter vsebina datoteke. Tudi ta je v obliki znakovnega niza.

4.1.4 Funkcionalnosti aplikacije

Razred *FragmentHome* je prva (naslovna) stran, ko aplikacijo poženemo. Tukaj so prikazani podatki univerze, izdajatelja izkaznice. Aplikacija ima tri glavne funkcionalnosti:

- pridobitev podatkov študentske izkaznice na mobilno napravo,
- prikaz izkaznice na zaslonu in
- identifikacijo izkaznice z bralnikom izkaznic NFC.

Za te funkcionalnosti skrbijo dva glavna razreda, *FragmentGetData* in *FragmentStudentId*, ter storitev *StudentCardService*. Vse razrede skupaj povezuje aktivnost *MainActivity*. Ta skrbi za življenjski cikel aplikacije in za to, kateri fragment je trenutno prikazan.

Pridobitev podatkov

Za pridobitev podatkov s strežnika skrbi razred *FragmentGetData*. Najprej smo poskrbeli, da vsi elementi (gumbi in vnosna polja) delujejo. Nato smo se lotili programiranja metod za pridobitev podatkov. Proces poteka v ozadju, kot je to večinoma standard pri velikem številu mobilnih aplikacij. To pomeni, da med tem procesom aplikacija deluje normalno in se ob morebitnih težavah ne ustavi.

Zgoraj opisane zahteve dosežemo z uporabo razreda *AsyncTask*. Ta poskrbi, da se koda, vstavljena vanj, izvaja vzporedno na ločeni niti, kot ločen proces, in ne moti delovanja glavnega procesa (delovanja same aplikacije). V tem razredu so uporabljene tri metode. Prva, *onPreExecute*, se izvede, preden začnemo pridobivati podatke. Tukaj prikažemo pojavno okno, ki nam pove, da se podatki pridobivajo. Nato sledi metoda *doInBackground*, ki poskrbi, da se zahteva po podatkih pošlje na spletno storitev ali sporoči napako. Napaka nastane, če nimamo dostopa do interneta. To napako moramo ujeti in o njej obvestiti uporabnika. Tretja metoda je *onPostExecute*, ki se izvede, če se je prejšnja metoda pravilno izvedla. Ob uspešni izvedbi vseh treh metod razreda *AsyncTask* uporabnika obvestimo, da so podatki uspešno shranjeni na mobilno napravo, in se prikaže prva stran aplikacije.

Za pridobitev podatkov spletne storitve smo uporabili razred *HttpURLConnection* [17]. Ta se uporablja za pošiljanje in pridobivanje podatkov s spleta. Podatki so lahko poljubnega tipa in

poljubne dolžine. Odgovor s spletne storitve dobimo v formatu JSON. Le-tega pretvorimo v znakovni niz in ga shranimo v datoteko.

Prikaz izkaznice

Razred *FragmentStudentId* skrbi za prikaz podatkov študentske izkaznice na zaslonu. Na prvi strani aplikacije je postavljen gumb Podatki o študentu. Ob kliku nanj se odpre druga stran aplikacije, kjer so prikazani zgoraj omenjeni podatki. Preden lahko podatke prikažemo, jih moramo prebrati iz datoteke, shranjene na mobilni napravi.

Prebrani znakovni niz iz datoteke je v formatu JSON. S pomočjo pomožnega razreda *JSONParser* iz prebranega niza izvlečemo želene podatke (ime, priimek, vpisna številka, UID, status). Z njimi s pomočjo pomožnega razreda *Student* ustvarimo objekt istega tipa. Zdaj, ko imamo objekt tipa *Student*, je dostopanje do podatkov preprosto z metodami *getName*, *getSurname*, *getEnrolNum*, *getUID* in *isStatus*.

Identifikacija izkaznice

Preverjanje identitete z Bralnikom NFC mora delovati brez vpliva uporabnika. To pomeni, da aplikacije ni treba zagnati zato, da bi se preverjanje identitete izvedlo. To je mogoče z uporabo storitev v Androidu. Pri Android 4.4 KitKat in novejših obstaja storitveni razred *HostApduService*, ki se uporablja za osnovo pri implementaciji načina emulacije kartic NFC [22]. Zato razred *StudentCardService* razširi zgoraj omenjeni razred in uporablja njegovi metodi *processCommandApdu* in *onDeactivated*.

Ko se z mobilno napravo, kjer je nameščena aplikacija Emulacija kartice NFC, približamo Bralniku NFC, le-ta mobilni napravi pošlje signal s številko AID. Možni sta dve opciji:

- Če se ta ujema s številko AID v aplikaciji na mobilni napravi, potem se izvede metoda *processCommandApdu*. Znotraj metode iz datoteke s podatki študentske izkaznice preberemo UID izkaznice. Le-ta se iz znakovnega niza pretvori v bitno obliko, nato se pošlje bralniku.

- Če se AID aplikacije Bralnik NFC in AID aplikacije Emulacija kartice NFC ne ujemata, se izvede metoda *onDeactivated*. Ta se uporablja za sporočanje napak, če se je izgubila povezava med napravama ali če je AID nepravilen.

4.2 Aplikacija za branje študentske izkaznice

Aplikacije za branje študentske izkaznice Bralnik NFC se uporablja za:

- pridobitev UID iz aplikacije Emulacija kartice NFC,
- preverjanje istovetnosti študenta s seznama na napravi ali v bazi podatkov na strežniku in
- prikaz podatkov študenta, kateremu pripada dobljeni UID.

Začeli smo z razvojem uporabniškega vmesnika in nato nadaljevali z razvojem storitve za branje emulirane izkaznice. Ob tem je bilo realizirano tudi shranjevanje podatkov o študentih v lokalno podatkovno bazo na mobilni napravi, kjer teče aplikacija. Dodali smo še možnost prikaza seznama študentov, shranjenih v lokalni podatkovni bazi. Nato pa je sledila vzpostavitev povezave s spletno storitvijo.

4.2.1 Arhitektura aplikacije

Arhitektura aplikacije je podobna kot pri aplikaciji, ki emulira študentsko izkaznico. Sestavljajo jo mape *java*, *res* ter *manifest*. Aplikaciji sta si oblikovno zelo podobni, tako da se vsebina mape *res* ne razlikuje veliko. Razlikujeta pa se vsebini map *java* in *manifest.*

V mapi *java* se nahajajo:

ena aktivnost (ang. Activity):

- MainActivity,

mapa *fragments* s tremi fragmenti (ang.fragments):

- CardReaderFragment,
- FragmentDialogExitApplication in
- ListViewDialogFragment,

mapa *database* z razredom:

- DatabaseConnector,

mapa *utils*, ki vsebuje pomožne razrede:

- JSONParser,
- Student,
- Utilities in
- ListViewAdapterStudent,

in ena storitev (ang. service):

- StudentCardReader.

V izvedbi je uporabljen en glavni fragment, ker aplikacija prikaže samo eno stran.

4.2.2 Uporabniški vmesnik

Uporabniški vmesnik je razvit enako kot v aplikaciji, ki emulira študentsko izkaznico. Prav tako je oblikovan s pomočjo atributov uteži, deluje v portretnem načinu in barve so enake. Razlika je v vsebini prikazanih podatkov. Druga razlika je, da ni interaktivnih gumbov, ker ves zaslon uporabimo za prikaz podatkov študenta, ter v spremembi velikosti besedila.

Predvidevali smo, da bi se ta aplikacija lahko poganjala tudi na tablicah, ki imajo zaslone občutno večje kot mobilni telefoni. V opravilni vrstici se nahajata dva gumba za prikaz seznama študentov. V njuno obliko se nismo poglabljali, saj nista del glavnega prikazovalnega fragmenta. Več pozornosti smo namenili funkcionalnostma, ki ju sprožita.

Lastnosti besedila so shranjene v datoteko *styles.xml*, ki se nahaja v podmapi *values* v mapi *res*. Tukaj določamo velikost, tip in obliko pisave ter vse druge lastnosti, ki oblikuje besedilo. Pri izpisu podatkov študenta (ime, priimek …) imamo na primer dve besedili, ki imata enake vse lastnosti, razen sloga pisave:

- Besedilo na levi strani (opisi polj za ime, priimek …) je velikosti 16, črne barve ter debeline krepko. Slog je poimenovan »text_view_16_bold«.
- Besedilo na desni strani (polja ime, priimek …) je tudi velikosti 16, črne barve, vendar pa ima debelino normalno. Ta slog smo shranili kot nov slog z imenom »text_view_16_normal«.

Tako ni treba za vsak element z besedilom opisovati vseh lastnosti posebej. Na začetku določimo vse sloge, ki jih nameravamo uporabiti, in si s tem olajšamo poznejše delo.

Če želimo uporabiti enak slog besedila in prilagoditi samo velikost besedila glede na velikost zaslona, zadevo lahko rešimo zelo preprosto. V mapi *res* ustvarimo več podmap *values*. Vsaki podmapi *values* pripišemo številko, ob kakšni velikosti zaslona naj se uporabi datoteka *styles.xml*, ki se nahaja v njej. Velikosti so v grobem omejene na:

- 320dp (podmapa values-sw320dp),
- 480dp (podmapa values-sw480dp),
- 600dp (podmapa values-sw600dp),
- 720dp (podmapa values-sw720dp) in
- 820dp (podmapa values-sw800dp).

Te številke pomenijo, kakšno gostoto neodvisnih pikslov (density-independet pixels - dp) lahko naprave prikažejo na zaslonu [21]. Imena slogov so v vseh podmapah enaka. Spremenjena je samo velikost besedila. Za primer vzemimo zgoraj omenjeni slog

»text_view_16_normal«. V podmapi values-sw480dp ima ta slog velikost besedila 16 in se uporabi na napravah z zaslonom z gostoto 480dp. V podmapi values-sw600dp pa ima slog z istim imenom velikost besedila 26. Ta velikost besedila se bo uporabila, ko se bo aplikacija izvajala na napravah, kjer je zaslon z gostoto 600dp. Vse druge lastnosti besedila pa ostanejo enake.

4.2.3 Podatkovna baza in prikaz seznama študentov

V opravilni vrstici sta dva gumba, Prikaži seznam in Pridobi podatke iz baze. Prvi prikaže seznam študentov, shranjenih v lokalni podatkovni bazi. Drugi gumb pošlje zahtevo po vseh študentih, shranjenih v podatkovni bazi, ki je povezana s spletno storitvijo, in z dobljenimi podatki napolni tabelo v lokalni podatkovni bazi.

Podatkovna baza

Podatkovna baza v mobilni napravi vsebuje tabelo, kjer so shranjeni podatki študentov. V ta namen je uporabljen razred *DatabaseConnector*. V njem so definirani stolpci, ki tvorijo tabelo podatkov, definirane so metode za vstavljanje novega zapisa oziroma študenta (*insertStudent*), za branje podatkov določenega študenta iz baze (*getStudent*) ter metoda, ki vrne seznam vseh študentov, shranjenih v bazi podatkov (*getListOfStudents*). Prav tako so definirane metode, ki bazo pripravijo za branje (*openToRead*), pisanje (*openToWrite*) ter zapiranje dostopa do baze po določeni operaciji (*close*). V razredu *DatabaseConnector* pa se uporablja še notranji razred *CardReaderDbHelper*, ki je podrazred razreda *SQLiteOpenHelper*, ki omogoča uporabo metode za ustvarjanje baze podatkov *onCreate* [20].

Prikaz seznama študentov v lokalni bazi podatkov

Za prikaz seznama študentov, shranjenih v bazi podatkov, sta uporabljena dva razreda, *ListViewDialogFragment* in *ListViewAdapterStudent*. Naloga prvega je, da se seznam prikaže kot pojavno plavajoče okno, drugi pa deluje kot polnilec seznama, kar pomeni, da napolni element za prikaz seznama *listView* s podatki.

V seznamu so prikazani trije osnovni podatki študenta: ime, priimek in vpisna številka. Vsaka vrstica v seznamu je interaktivna, kar pomeni, da se ob kliku nanjo na glavnem prikazovalnem zaslonu aplikacije prikažejo vsi podatki izbranega študenta.

4.2.4 Funkcionalnosti aplikacije

Aplikacija ima dve glavni funkcionalnosti, branje UID emulirane študentske izkaznice in prikaz podatkov študenta, kateremu pripada izkaznica s prejetim UID. Branje UID opravlja razred *StudentCardReader*, prikaz podatkov študenta, kateremu pripada prejeti UID, pa izvaja razred *CardReaderFragment*.

Branje UID emulirane izkaznice NFC

Branje kartice NFC je mogoče le, ko je aplikacija zagnana, ima stanje branja (*readerMode*) omogočeno ter ima naprava vključen NFC. Stanje branja se spremeni, če je glavni proces aplikacije zaseden oziroma začasno ustavljen. Razred *StudentCardReader* implementira vmesnik *NfcAdapter.ReaderCallback*, ki aplikaciji omogoča spremembo bralnega stanja [18]. Razred vsebuje metodo *onTagDiscovered*, ki se izvede, ko je odkrita nova značka NFC. Če želi aplikacija komunicirati z mobilno napravo, ki uporablja HCE, je treba odkrito značko procesirati s pomočjo razreda *IsoDep* [19]. Z objektom tipa *IsoDep* se preveri, ali značka podpira ISO-DEP. Če je odgovor pritrdilen, se metoda izvaja naprej, če je nikalen, pa se ustavi.

Naslednji korak je pošiljanje številke AID, ki je shranjena v aplikaciji, mobilni napravi, ki emulira študentsko izkaznico. Če se AID-številka aplikacije ujema z AID-številko emulirane izkaznice, aplikacija prejme UID od prebrane študentske izkaznice. Sledi klicanje metode *onAccountReceived*, ki je definirana v razredu *CardReaderFragment*, ta prikaže podatke študenta s številko UID.

Prikaz podatkov

Razred *CardReaderFragment* omogoča prikazovanje podatkov študenta na zaslonu. Ko razred *StudentCardReader* opravi svojo nalogo in je UID izkaznice prejet, aplikacija najprej preveri, ali ima študenta z izkaznico, kateri pripada omenjeni UID, shranjenega v lokalni podatkovni bazi. Če ga ima, potem se pokliče metoda *onAccountReceived*, ki njegove podatke prikaže na zaslonu. Če študenta z iskanim UID ni v bazi podatkov na telefonu, pa aplikacija pošlje spletni storitvi zahtevo z ustreznim UID. Ko spletna storitev odgovori in vrne podatke o študentu, aplikacija podatke študenta shrani v lokalno podatkovno bazo ter jih prikaže na zaslonu. Ob negativnem odgovoru spletne storitve pa aplikacija sporoči uporabniku, da študenta ni v podatkovni bazi. Za pridobitev podatkov spletne storitve je uporabljen razred *HttpURLConnection*.

4.3 Razvoj spletne aplikacije

Spletna aplikacija omogoča ustvarjanje podatkovne baze o študentih. Poleg tega omogoča tudi prikaz ter dodajanje, brisanje in urejanje podatkov teh študentov, kot tudi prikaz seznama vseh zapisov v podatkovni bazi. Mobilnima aplikacijama omogoča dostop do podatkov študentov, ki so shranjeni v podatkovni bazi na strežniku.

4.3.1 Priprava strežnika in podatkovne baze

Spletna storitev in baza podatkov se nahajata na platformi Microsoft Windows Azure. Za bazo podatkov je bila uporabljena podatkovna baza Microsoft SQL. Sestavljata jo dve tabeli, kjer so v eni shranjeni podatki študenta, ki so zapisani na študentski izkaznici (ime, priimek, vpisna številka, status in UID), v drugi pa podatki uporabnikov spletne aplikacije. V tabelo študentov smo dodali še stolpec, kjer je zapisano geslo, ki ga študent uporabi, ko pošlje zahtevo za pridobitev podatkov študentske izkaznice na svojo mobilno napravo. Na platformi Azure smo ustvarili še prazno spletno stran, ki bo omogočala gostovanje spletne storitve RESTful.

4.3.2 Arhitektura aplikacije

Spletna aplikacija je bila razvita s pomočjo orodja Visual Studio 2012. Windows Azure SDK za Visual Studio nam omogoča, da podatkovno bazo uporabljamo že med razvojem aplikacije, tako ni treba aplikacije vsakič znova objavljati na spletni strani in preverjati, ali povezava z bazo podatkov deluje.

Izvedba spletne aplikacije je potekala z orodjem ASP.NET MVC, kar nam omogoča popolno kontrolo nad vsakim delom aplikacije. Najpomembnejši so krmilniki (ang. Controller), saj z njimi upravljamo podatke v modelu (ang. Model) in jih po želji prikazujemo v prikazovalnih straneh (ang. View).

Z ADO.NET Entity Data Model ustvarimo model za dostop do podatkovne baze. Za tabelo v bazi kreiramo krmilnik, prek katerega bomo z mobilnima aplikacijama dostopali do podatkovne baze. Prav tako kreiramo krmilnik za prikaz podatkov iz baze na spletni strani. Nato ustvarimo še prikazovalne strani, kjer bomo prikazali seznam študentov iz baze, ter obrazec za dodajanje novih in urejanje že obstoječih študentov v bazi podatkov.

Aplikacijo v Visual Studiu sestavljajo krmilniki v mapi Controllers:

- DodajController.cs (razred z metodami za dodajanje novega študenta),
- HomeController.cs (razred omogoča uporabniku dostop do spletne aplikacije),

- StudentCardController.cs (krmilnik API, namenjen mobilnim aplikacijam za dostop do podatkov),
- StudentiController.cs (razred se uporablja za upravljanje podatkov na spletni strani),

model v mapi Models:

- ModelED, ki vsebuje razreda Student in Login,

ter prikazovalne strani:

- DodajStudenta.cshtml, v kateri se nahaja obrazec za dodajanje novega študenta,
- List.cshtml, kjer je prikazan seznam vseh študentov, vpisanih na fakulteto,
- Edit.cshtml, v katerem lahko spremenimo podatke izbranega študenta,
- Login.cshtml, namenjena za vnos uporabniškega imena in gesla uporabnika spletne storitve, in
- Layout.cshtml, ki povezuje vse prikazovalne strani skupaj.

Dostopnost podatkov mobilnim aplikacijam

Mobilni aplikaciji za svoje delovanje potrebujeta dostop do podatkov študentov v bazi podatkov na strežniku. Za to skrbi razred *StudentCardController*. To je najpomembnejši razred spletne aplikacije v povezavi z mobilnima aplikacijama. Ta vsebuje metode, ki se na podlagi prejetih parametrov odzivajo na zahteve http.

Metoda *GetStudentData* ima vhodna parametra vpisno številko in geslo, s katerima poišče podatke študenta s to vpisno številko in odgovori s podatki tega študenta. Ti podatki so v formatu JSON. Tako zahtevo pošlje aplikacija, ki emulira študentsko izkaznico, ko le-to želimo pridobiti na mobilno napravo.

Metoda *GetStudentDataByUID* je metoda, ki jo kliče aplikacija za branje izkaznic NFC. Zahtevo s temi podatki dobi, ko aplikacija ne najde študenta z določeno izkaznico, ker ni zapisan v lokalni bazi podatkov. Ko ga najde, njegove podatke pošlje aplikaciji. Če pa ga ne najde, to sporoči aplikaciji.

Metoda *GetStudentis* vrača seznam vseh študentov, shranjenih v bazo podatkov.

4.4 Testiranje aplikacij

Aplikacijo Emulacija kartice NFC uporabljamo na dva načina:

- Prikaz podatkov o študentu, ki so enaki kot je natisnjena vsebina na študentski izkaznici NFC.

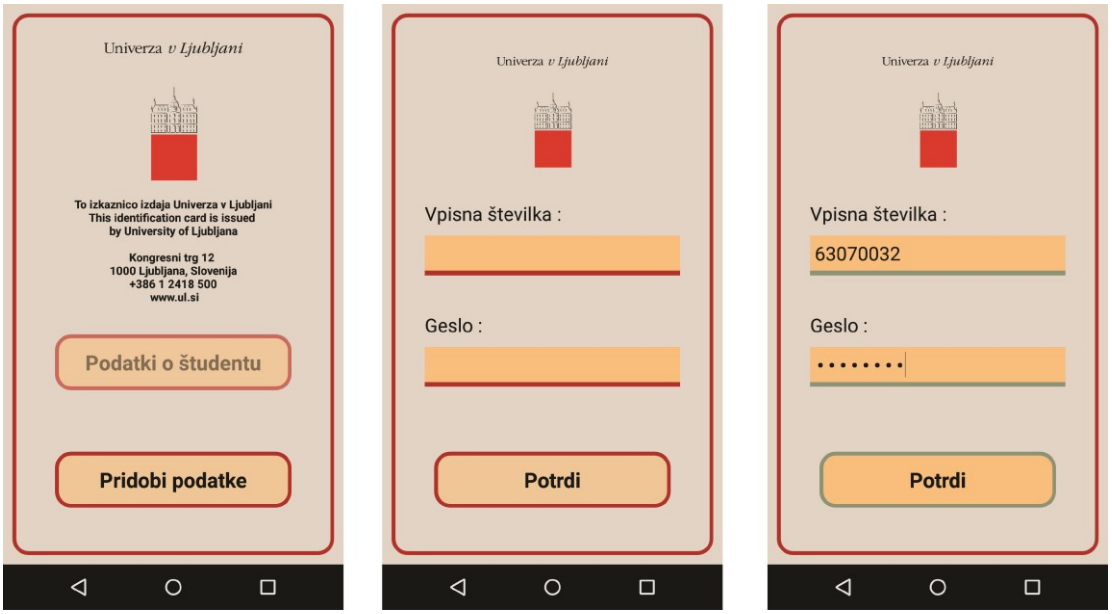
- Preverjanje identitete študenta poteka tako, da pametni telefon prislonimo na bralnik študentske izkaznice Bralnik NFC. Po prejetem signalu iz bralnika aplikacija pošlje UID (svojo identifikacijsko številko). Bralnik NFC preveri, ali ima študenta, kateremu pripada izkaznica z iskanim UID, v lokalni bazi podatkov. Če zapisa ni, preveri še v bazi podatkov na strežniku. Če najde podatke, jih prikaže na zaslonu.

4.4.1 Postopek pridobitve in uporaba emulirane študentske izkaznice NFC na pametnem telefonu

Aplikacije Emulacija kartice NFC ni treba zagnati vsakič, ko jo prislonimo na bralnik. Moramo pa telefon odkleniti, če jo želimo uporabiti ter imeti vklopljen NFC.

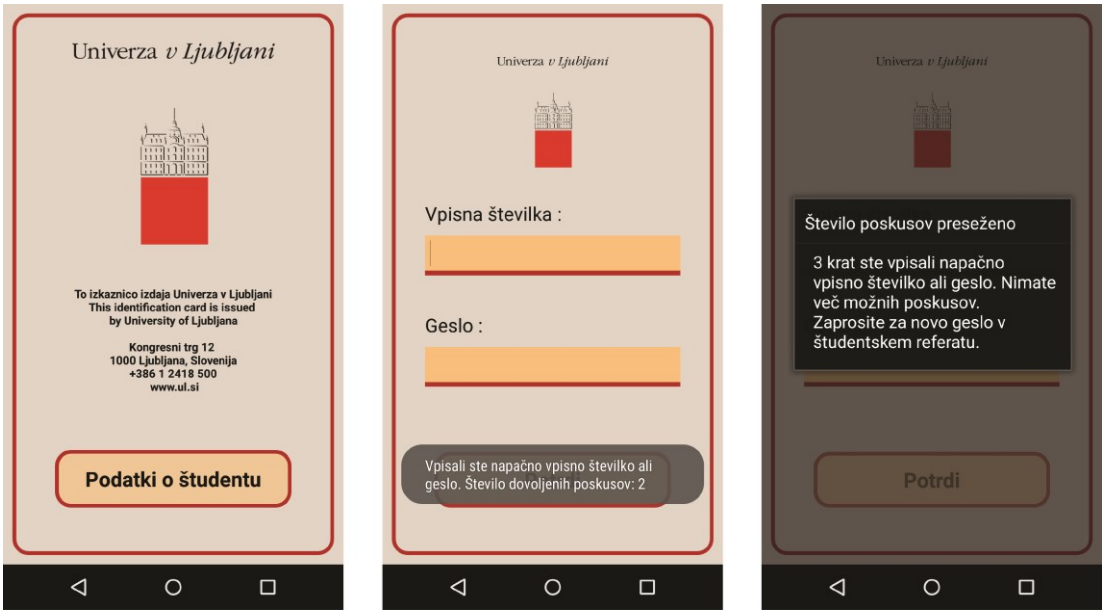
Postopek pridobitve emulirane študentske izkaznice NFC:

- Aplikacijo Emulacija kartice NFC najprej naložimo na mobilno napravo. Za namestitev sta potrebni vpisna številka in geslo, ki ju dobi študent ob vpisu na fakulteti.
- Ob prvem zagonu se prikaže začetni zaslon, kjer so prikazani logotip univerze, podatki izdajatelja izkaznice ter dva gumba (slika 6 levo). Zdaj lahko izberemo samo gumb z napisom Pridobi podatke, z možnostjo pridobitve študentske izkaznice na mobilno napravo. Drugi gumb, Podatki o študentu, je neaktiven.
- Ob kliku na gumb Pridobi podatke se odpre novo okno, kjer imamo dve tekstovni polji, Vpisna številka in Geslo (slika 6 sredina). V prvo vpišemo svojo vpisno številko, v drugo pa geslo.
- Kliknemo gumb Potrdi (slika 6 desno).



Slika 6: Začetni zaslon (levo), zaslon za vpis podatkov (na sredini), gumb Potrdi (desno) za povezavo z bazo.

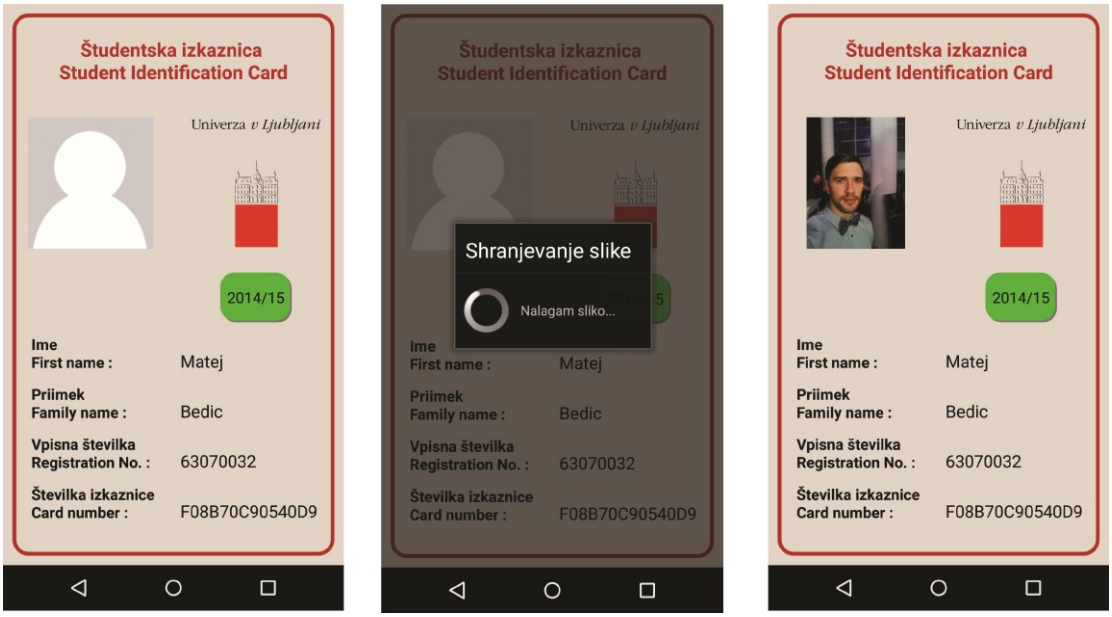
- Na spletno aplikacijo se pošlje zahteva z vpisanima podatkoma za pridobitev študentske izkaznice.
- Če so bili vneseni podatki pravilni in smo na mobilno napravo dobili študentsko izkaznico, nas aplikacija vrne na začetni zaslon (slika 7 levo). Zdaj gumb za pridobitev podatkov ni več prikazan, saj imamo podatke že na telefonu.
- Če smo vpisali napačno vpisno številko ali geslo, nas aplikacija o tem obvesti, izprazni vpisana polja ter omogoči nov poskus (slika 7 na sredini). Imamo tri možnosti za vpis in če so vsi poskusi napačni, nas aplikacija obvesti, da nimamo več možnosti poskusiti (slika 7 desno). Nato se aplikacija zaklene, vnosna polja in gumb Potrdi postanejo neaktivni in aplikacijo moramo znova naložiti. Pred tem moramo fakulteto obvestiti in prositi za novo geslo.



Slika 7: Začetna stran po pravilnem vnosu (levo), obvestilo ob napačnem vnosu podatkov (na sredini), obvestilo o prekoračitvi poskusov vnosov podatkov (desno).

Uporaba emulirane študentske izkaznice NFC:

- Gumb Prikaži izkaznico odpre novo okno, kjer so prikazani podatki študentske izkaznice (ime, priimek, vpisna številka, status študenta, številka izkaznice, logotip Univerze v Ljubljani) (slika 8 levo). Levo zgoraj je tudi prostor, kamor lahko shranimo sliko, ki si jo izberemo v skladu z zahtevami iz svoje galerije (slika 8 na sredini in desno). Na prejšnji zaslon se vrnemo s pritiskom gumba za nazaj, ki je na mobilni napravi.
- Status študenta je prikazan kot neaktiven gumb. Če je obarvan zeleno z zapisom letnice tekočega študijskega leta, to pomeni, da študent ima status in je redno vpisan, če je obarvan rdeče, brez besedila, pa pomeni, da študent nima statusa.
- Za izhod iz aplikacije se moramo vrniti na začetni zaslon in nato znova pritisniti gumb Nazaj na telefonu. Tedaj se prikaže okno, kjer nas vpraša, ali želimo aplikacijo zapreti.



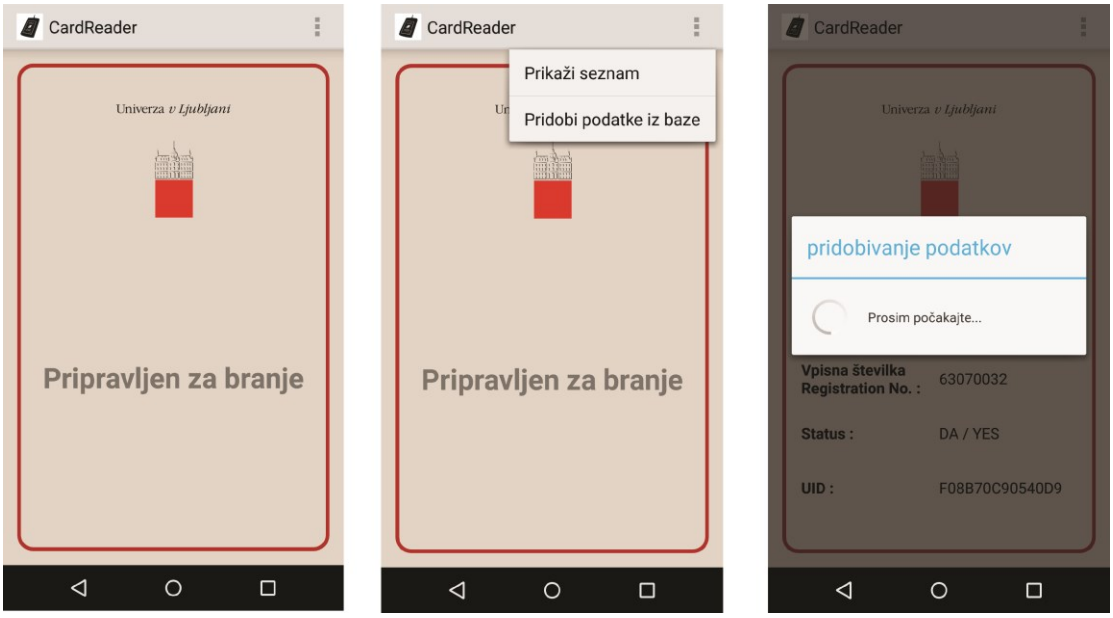
Slika 8: Prikaz podatkov o študentu (levo), shranjevanje izbrane slike (na sredini), podatki o študentu z izbrano sliko (desno).

4.4.2 Aplikacija za branje študentskih izkaznic

Aplikacija za branje študentske izkaznice je testna aplikacija, namenjena za preverjanje identitete študentov. Preverjanje se opravlja s podatki v bazi podatkov na strežniku. Preverjanje, ali je študent shranjen v bazi podatkov, na telefonu ali strežniku, se opravlja na podlagi UID, ki ga bralnik prejme iz mobilne naprave, kjer je shranjena izkaznica. Preverjanje je avtomatično. To pomeni, če podatki študenta z iskanim UID niso shranjeni na telefonu, aplikacija sama preveri, ali so podatki na strežniku.

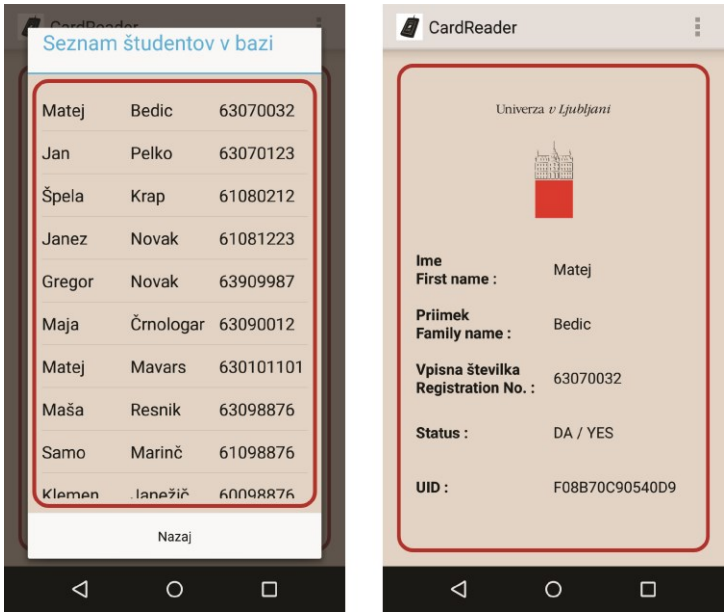
Uporaba aplikacije Bralnik NFC:

- Aplikacija je pripravljena za uporabo takoj po namestitvi.
- Ob prvem zagonu je podatkovna baza v aplikaciji prazna.
- Desno zgoraj se nahaja gumb, ki prikaže podmeni z dvema gumboma, Prikaži seznam in Pridobi podatke iz baze. Prvi prikaže seznam študentov, ki so shranjeni v bazi na telefonu. Drugi pa pridobi podatke iz podatkovne baze na strežniku (slika 9 na sredini).



Slika 9: Začetni zaslon (levo), prikaz podmenija (na sredini), pridobivanje podatkov (desno).

- Ko preberemo študentsko izkaznico z drugega telefona in če je študent vpisan v bazo podatkov na telefonu ali na strežniku (v tem primeru potrebujemo spletno povezavo), se nam podatki takoj izpišejo (slika 10 desno). Drugače se podatki ne prikažejo.
- Po branju izkaznice se na začetni zaslon lahko vrnemo ob kliku na gumb Nazaj na telefonu (slika 9 levo).
- Za naslednje branje izkaznice se ni treba vračati na začetni zaslon.
- Če želimo podatke določenega študenta znova videti, lahko kliknemo nanj v seznamu, ki se nam prikaže ob kliku na gumb Prikaži seznam (slika 10 levo). Tako se nam izpišejo vsi podatki izbranega študenta.



Slika 10: Prikaz seznama študentov v lokalni bazi podatkov (levo), prikaz podatkov študenta po branju emulirane študentske izkaznice (desno).

- Če študenta, čigar izkaznico smo prebrali, ni v bazi podatkov na telefonu, aplikacija avtomatično pošlje zahtevo o podatkih študenta na strežnik spletne aplikacije. Če podatke prejme, jih shrani v bazo podatkov na telefonu in jih prikaže na zaslonu.
- Če študenta ni v bazi podatkov, na telefonu in v bazi podatkov na spletu, nam aplikacija sporoči, da študent ni vpisan v bazo podatkov.

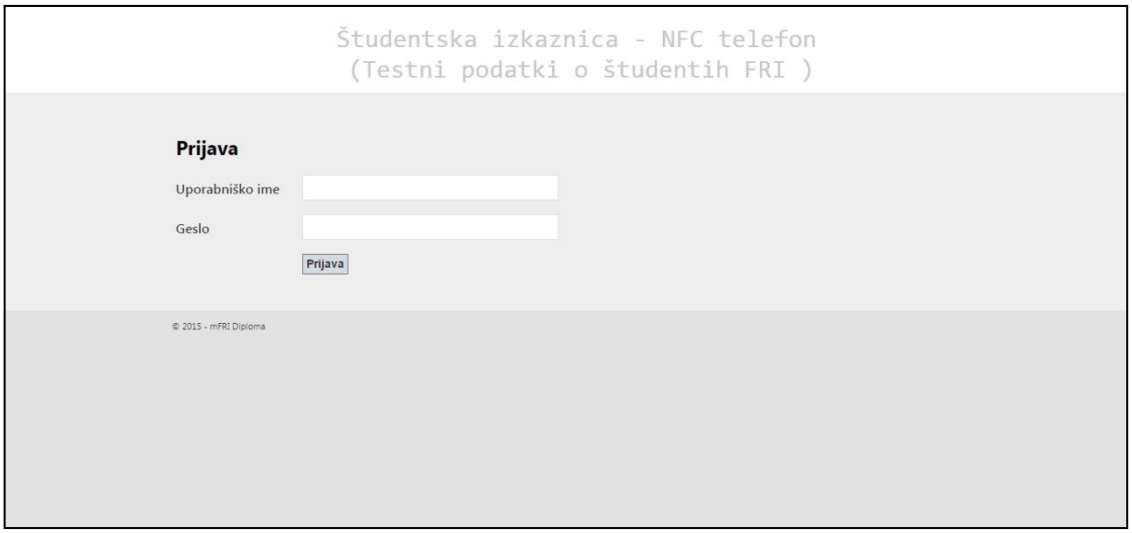
Za izhod iz aplikacije se moramo vrniti na začetni zaslon in nato znova pritisniti gumb Nazaj na telefonu. Tedaj se prikaže okno, kjer nas vpraša, ali želimo aplikacijo zapreti.

4.4.3 Spletna aplikacija z bazo podatkov

Spletna aplikacija za testiranje nadomešča študijski informacijski sistem in jo obe mobilni aplikaciji uporabljata za pridobitev podatkov o študentih v svojo datoteko oziroma podatkovno bazo. Za lažje testiranje je v seznamu študentov prikazano tudi polje z geslom, ki ga potrebujemo, če želimo pridobiti študentsko izkaznico NFC na telefon.

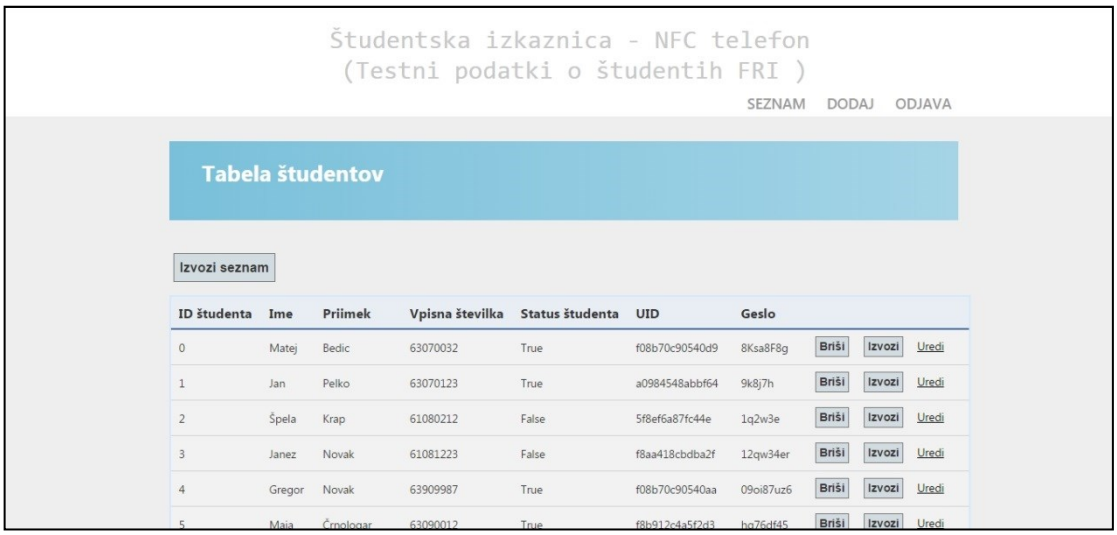
Uporaba spletne aplikacije je zasnovana tako, da omogoča ustvarjanje podatkovne baze študentov in nato preverjanje njihovih podatkov:

- Ob prvem zagonu se prikaže okno za vpis uporabnika. Tukaj vpišemo uporabniško ime in geslo (slika 11) . Tako podatki na strežniku niso prosto dostopni.



Slika 11: Začetna stran spletne aplikacije.

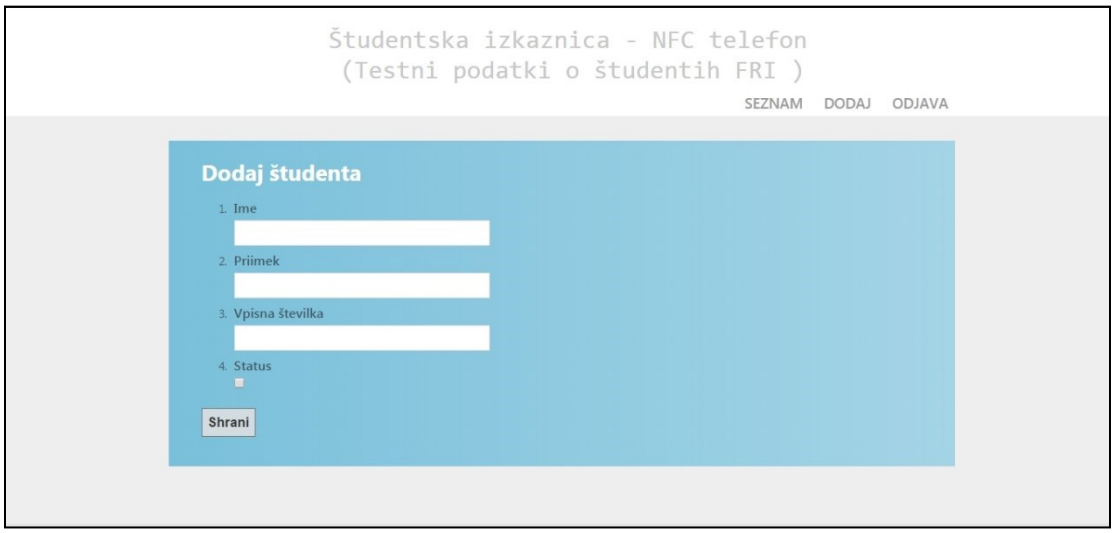
- Nato se nam prikaže seznam študentov, katerih podatki so shranjeni v bazi podatkov na strežniku (slika 12). Na vrhu seznama je gumb Izvozi seznam, ki omogoča izvoz v datoteko .csv.
- Pri seznamu imamo tudi možnost brisanja in spreminjanja podatkov vsakega študenta ter izvoz njegovih podatkov v datoteko .csv.



Slika 12: Seznam študentov v bazi podatkov na strežniku.

- Če kliknemo gumb Briši, nas brskalnik vpraša, ali smo prepričani. Če odgovorimo pritrdilno, se podatki tega študenta izbrišejo. Če odgovorimo nikalno, ostaneta baza podatkov in seznam študentov nespremenjena.

- Če kliknemo na gumb Uredi, se nam odpre novo okno, kjer so v tekstovna polja vpisani podatki izbranega študenta. Če jih popravimo, kliknemo gumb Shrani in podatki se v bazi podatkov in seznamu popravijo.
- V meniju desno zgoraj imamo tri gumbe: SEZNAM, DODAJ in ODJAVA. Prvi prikaže seznam študentov, drugi odpre novo podstran, kjer se prikaže obrazec za dodajanje novega študenta.
- Ob dodajanju novega študenta moramo vpisati ime, priimek, vpisno številko in določiti, ali ima status ali ne (slika 13). Številko UID izkaznice in geslo, ki ga bo študent uporabil za pridobitev izkaznice na svojo mobilno napravo, se avtomatsko generirata. Vsako vnosno polje ima določen format zapisa. Polje za vnos imena in priimka lahko vsebuje samo črke (veliko začetnico, ki ji sledijo male črke). Polje za vpisno številko lahko vsebuje le številke. Dolžina vpisne številke je 8 ali 9 znakov. Ko kliknemo gumb Shrani, se podatki zapišejo v bazo podatkov, vsa polja za vnos pa se izbrišejo in tako imamo možnost dodajanja novega študenta.



Slika 13: Obrazec za vnos novega študenta v bazo podatkov.

- Zadnji gumb v meniju pa aplikacijo zapre in znova odpre podstran za vpis uporabnika (slika 11).

Poglavje 5 Sklep

V diplomskem delu smo predstavili in razvili aplikacijo, ki emulira študentsko izkaznico NFC v pametnem telefonu. Aplikacijo bi lahko uporabljal vsak študent, ki ima pametni telefon z operacijskim sistemom Android 4.4 KitKat ali novejšim. Telefon mora prav tako imeti možnost uporabe tehnologije NFC.

Za testiranje smo razvili še aplikacijo za branje kartic NFC in spletno aplikacijo z bazo podatkov. Z mobilno aplikacijo za branje kartic NFC smo testirali komunikacijo NFC med mobilnima napravama. Spletna aplikacija nam je zagotovila pridobivanje podatkov študentov, saj pomeni testni sistem s podatkovno bazo, kjer so shranjeni podatki študentov, vpisanih na fakulteti.

Emulirana študentska izkaznica NFC je zelo enostavna za uporabo. Namestitev je hitra in preprosta in ko je enkrat aplikacija nameščena, je za uporabo ni treba vsakič zagnati. Vse, kar je potrebno za delovanje, je, da imamo vklopljen NFC. Ko se želi študent identificirati bralniku kartic NFC, mora le odkleniti svoj telefon, na katerem teče emulacija izkaznice, in ga približati bralniku. Bralnik kartic NFC bi uporabljal profesor, ki bi z njim lahko preverjal prisotnost na predavanjih. Seveda bi moral ob identifikaciji priložiti tudi osebni dokument. Aplikacija ima tudi možnost prikaza podatkov študentske izkaznice na zaslonu telefona. V tem primeru pa jo študent zažene na svojem telefonu in na zaslonu se prikažejo njegovi podatki, vključno s statusom vpisa.

Rešitev se nam zdi zelo zanimiva in uporabna. Obstajajo tudi številne možnosti za nadgradnjo. Možna bi bila uporaba emulirane študentske izkaznice NFC pri izposoji knjig v knjižnici na fakulteti ali pri elektronskem dostopu do določenih prostorov na fakulteti (na primer čitalnic in podobno).

LITERATURA

[1] Vedat Koskun, Kerem Ok, Busra Ozdenizci, »Professional NFC Application Development for Android«, 2013.

[2] Jess Chadwick, Todd Snyder, Hrusikesh Panda, »Programming ASP.NET MVC 4«, 2012.

[3] (2014) Pravilnik o študentski izkaznici. Dostopno na: http://www.uni-lj.si/o_univerzi_v_ljubljani/organizacija__pravilniki_in_porocila/predpisi_statut_ul_in_pravilniki/20130711162210 92/.

[4] (2014) Kriptirni algoritmi DES, 3DES in AES. Dostopno na: http://arxiv.org/ftp/arxiv/papers/1003/1003.4085.pdf.

[5] (2014) Near Field Communication. Dostopno na: http://developer.android.com/guide/topics/connectivity/nfc/index.html.

[6] (2014) MF3ICD81, MF3ICD41, MF3ICD21. Dostopno na: http://www.acs.com.hk/download-manual/2266/TDS DESF EV1.pdf.

[7] (2015) REST in RESTfull. Dostopno na: https://mauriziostorani.wordpress.com/2008/07/27/rest-representational-state-transfer-and-restful-web-services-concepts-and-examples/.

[8] (2014) Host-based Card Emulation. Dostopno na: http://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[9] (2015) ISO/IEC 7816-4 standard. Dostopno na: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4.aspx.

[10] (2015) Izkaznica NFC v postopku preverjanja študijskih obveznosti, Dean Koštomaj. Diplomsko delo. Dostopno na: http://eprints.fri.uni-lj.si/2732/1/63110295-DEAN_KO%C5%A0TOMAJ-Izkaznica_NFC_v_postopku_preverjanja_%C5%A1tudijskih_obveznosti.pdf.

[11] (2015) NFC Forum. Dostopno na:
http://nfc-forum.org/.

[12] (2015) Control your smartphone with NFC tags. Dostopno na:
http://the-gadgeteer.com/2013/03/11/control-your-smartphone-with-nfc-tags/.

[13] (2015) Android Studio overview. Dostopno na:
http://developer.android.com/tools/studio/index.html.

[14] (2015) SQLite. Dostopno na:
http://www.sqlite.org/about.html.

[15] (2015) Ciper class, Android Developers. Dostopno na:
http://developer.android.com/reference/javax/crypto/Cipher.html.

[16] (2015) Advanced Encryption Standard (AES). Dostopno na:
http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard.

[17] (2015) HttpURLConnection. Dostopno na:
http://developer.android.com/reference/java/net/HttpURLConnection.html.

[18] (2015) NfcAdapter.ReaderCallback. Dostopno na:
https://developer.android.com/reference/android/nfc/NfcAdapter.ReaderCallback.html.

[19] (2015) IsoDep. Dostopno na:
http://developer.android.com/reference/android/nfc/tech/IsoDep.html.

[20] (2015) SQLiteOpenHelper. Dostopno na:
http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html.

[21] (2015) Supporting Multiple Screens. Dostopno na:
http://developer.android.com/guide/practices/screens_support.html.

[22] (2105) HostApduService. Dostopno na:
http://developer.android.com/reference/android/nfc/cardemulation/HostApduService.html.

[23] (2015) Platforma Windows Azure. Dostopno na:
http://www.diventic.si/.

[24] (2015) .NET Framework. Dostopno na:
https://msdn.microsoft.com/en-us/library/zw4w595w.aspx.

[25] (2015) JSON. Dostopno na:
http://json.org/.

[26] (2015) NXP begins shipping NFC tags that can wake up a host device. Dostopno na:
http://www.nfcworld.com/2013/08/14/325492/nxp-begins-shipping-nfc-tags-that-can-wake-up-a-host-device.